J Supercomput
DOI 10.1007/s11227-011-0681-6

Mobile cloud computing framework for a pervasive and ubiquitous environment

Min Choi · Jonghyuk Park · Young-Sik Jeong

Abstract The increasing use of wireless Internet and smartphone has accelerated the need for pervasive and ubiquitous computing (PUC). Smartphones stimulate growth of location-based service and mobile cloud computing. However, smartphone mobile computing poses challenges because of the limited battery capacity, constraints of wireless networks and the limitations of device. A fundamental challenge arises as a result of power-inefficiency of location awareness. The location awareness is one of smartphone's killer applications; it runs steadily and consumes a large amount of power. Another fundamental challenge stems from the fact that smartphone mobile devices are generally less powerful than other devices. Therefore, it is necessary to offload the computation-intensive part by careful partitioning of application functions across a cloud. In this paper, we propose an energy-efficient location-based service (LBS) and mobile cloud convergence. This framework reduces the power dissipation of LBSs by substituting power-intensive sensors with the use of less-power-intensive sensors, when the smartphone is in a static state, for example, when lying idle on a table in an office. The substitution is controlled by a finite state machine with a user-movement detection strategy. We also propose a seamless connection handover mechanism between different access networks. For convenient on-site establishment,

M. Choi
School of Information and Communication Engineering, Chungbuk National University, Cheongju, Korea
e-mail: mchoi@cbnu.ac.kr

Y.-S. Jeong (✉)
Department of Computer Engineering, Wonkwang University, Iksan, Korea
e-mail: ysjeong@wku.ac.kr

J. Park
Department of Computer Engineering, Seoul National University of Science and Technology, Seoul, Korea
e-mail: parkjonghyuk1@hotmail.com

Published online: 23 September 2011

our approach is based on the end-to-end architecture between server and a smartphone that is independent of the internal architecture of current 3G cellular networks.

Keywords Low-power location awareness · Location-based service · Mobile cloud offloading · Connection handover · Pervasive and ubiquitous computing · Cloud computing

1 Introduction

The explosive growth in the use of the Internet has led to the development of mobile Internet. Nowadays, Mobile WiMAX and Wi-Fi facilitate convenient use of the Internet. Actually, the number of 802.16e (Mobile WiMAX) users is projected to increase to 80 million by 2013 [1]. The rising popularity of wireless Internet has triggered the spread of smartphones. Worldwide smartphone sales to end users totaled 1.6 billion units in 2010, a 31.8% increase from the figures in 2009 [2]. This rise in the use of wireless Internet and smartphones has accelerated the need for pervasive and ubiquitous computing.

Pervasive and ubiquitous computing (PUC) is the growing trend towards embedding microprocessors in everyday objects so that they can exchange information. The words "pervasive" and "ubiquitous" mean "existing everywhere." PUC devices are fully connected and constantly available. PUC relies on the convergence of wireless technologies, advanced electronics and the Internet. Today's mobile electronics are not just mobile communication devices; they also change people's lifestyles and create new cultures. Wherever users are located, the data can be found. In fact, things that people have not ever imagined before can be realized by pervasive and ubiquitous computing. In this paper, we introduce a mobile application framework for PUC. To realize PUC, the mobile application framework should involve a low-power location-based service and mobile cloud convergence.

- Energy-efficient location-based service: In PUC, mobility is the basis for everything. Therefore, we focus on conserving energy when we are using LBSs in mobile devices. This is because our mobile application framework is intended for PUC, which can involve frequent movements of mobile devices. This framework helps reducing power dissipation by substituting power-intensive sensors with less-power-intensive sensors when the smartphone is in a static state, for example, when lying idle on a table in an office.

Cloud computing [4] has emerged as a new computing paradigm that targets reliable and customizable services. It is a result of decades of research in virtual machine, distributed and parallel computing, utility computing, and more recently, networking, web service, and software as a service. In this study, we provide a seamless connection handover on mobile-cloud converged applications. This is useful for cloud computing environment in which many clients have mobility. With the wireless Internet facility, mobile users can move during Internet communication.

- Mobile cloud [5] convergence: As mentioned above, data distribution is the key issue for mobility. For mobile cloud convergence, task distribution is important,

because the computing power of mobile devices is not powerful enough for the devices to be the main computing platform. Mobile cloud convergence provides a solution to the computation power problem. The parts that need more computation run on the cloud, while the parts associated with the user interface run on the mobile device. IPC (inter-process communication) is one of the ways to realize this convergence. In this paper, we exploit this concept for PI value calculation. We improve the PI calculation algorithm by optimization for mobile cloud convergence.

- Seamless connection handover: There are certain problems in achieving a seamless connection handover, namely, communication failure and connection re-establishment. Communication channel flushing by zero window notification helps to resolve the communication failure problems. TCP port inheritance prevents connection re-establishment errors during socket reconstruction. Thus, our seamless connection handover technique is now able to preserve open network connections, and even for server sockets. This is a highly transparent approach, which neither introduces additional messages for channel flushing nor makes any modification to the TCP protocol stack. Experimental results show that the overhead due to the connection handover is almost negligible when compared with the time required by the conventional approach.

We implement the three design principles on a Samsung Galaxy S [6] Android smartphone as a middleware and evaluate the implementation extensively via measurements. While the proposed design principles are general enough to be applied to any software stack, the middleware implementation allows for better application transparency in the sense that applications can be kept as-is. We choose Android-based smartphones for prototyping because of the openness of the Android platform [3]. Our evaluation results with the implementation show that the proposed framework significantly saves energy in location sensing. The development framework of could convergence mobile applications allows these constructs to cooperate implicitly so that they create a synergy effect at the smartphone system level and successfully provide a framework at PUC environment. To summarize, the contributions of this work are as follows:

- We address and explore energy efficiency of GPS sensing for resource-constrained smartphones that often run location-based services (LBS).
- We propose three design principles tailored for LBS to reduce energy consumption in GPS sensing on smartphones and show that the integration of the proposed design principles leads to significant energy savings.
- We prototype the proposed design in Android-based smartphones, which are open to both practice and research, and demonstrate the effectiveness through real-life measurements.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 presents the key design principles of low-power location awareness. Section 4 describes our design and implementation of mobile cloud convergence. Section 5 shows the seamless connection handover mechanism in mobile cloud. We conclude by summarizing our results in Sect. 6.

2 Related work

A few prior pieces of work have attempted the location determination. Many applications using GPS for navigational purposes, particularly handheld applications, require low power in order to preserve battery life. The primary components that dissipate large amounts of power are the GPS. Moreover, the use of smartphones becomes pervasive. These smartphones are equipped with location sensing capability to enable LBS. Existing module platforms including Android do not employ techniques similar to our designs to improve power dissipation of LBS, although application developers partially adopt similar concepts. As users are increasingly adopting a wide variety of LBS on smartphones [7–9], since typical smartphones are equipped with multiple types of sensors, applications that take advantage of these sensors are booming, and many existing works attempt to detect an extract users' states and context based on the readings from these sensors [10–13]. Many approaches have been proposed to combine the information obtained from sensors including accelerometer, audio, GPS, camera, and so on [14–17]. Several pieces of work have attempted to learn mobility patterns. BreadCrumbs [20] uses the space history of a user to train a mobility model for each specific user and uses it to schedule network usage with the connectivity forecasts. Zheng et al. [21] use supervised learning to infer motion modes (e.g., walking, bus, driving) from their GPS logs. Sohn et al. [22] also recognize mobility.

According to the latest study from Juniper Research [23], the market for cloud-based mobile applications will grow 88% from 2009 to 2014. The market was just over $400 million this past year, says Juniper, but by 2014 it will reach $9.5 billion. Driving this growth will be the adoption of the new web standard HTML5, increased mobile broadband coverage and the need for always-on collaborative services for the enterprise. Several systems have been developed to support connection handoff at the user-level and can be run on unmodified commercial operating systems. These systems include Condor [24], CoCheck [25]. Since there is no kernel support for process migration, these systems require the processes not to use some common operating system services such as not allowing processes to fork or exec. Moreover, migration of networked applications is inherently impossible without kernel support. Kernel-level approaches for process migration include CRAK [26]. Solaris MC [27] provides connection migration with socket support. However, CRAK is only able to migrate the process which operates on client socket, not server socket. MOSIX, OpenSSI and Solaris MC provide migration by redirecting location-dependent operations such as system calls or socket connections. This approach degrades performance, fault-resilience and adversely affects reliability, because a migrated foreign process will still depend on its home node.

3 Power-saving strategy for smartphones

There have been concentrated efforts to reduce power dissipation in handheld devices, because these devices are battery-powered. Smartphone users are particularly conscious of battery exhaustion and the need for battery-saving techniques. To gather basic information for an energy-efficient strategy with regard to smartphone resource

management, this paper analyzes the power dissipation behavior of smartphone with regard to applications and functionalities [18]. From this analysis, we propose a low-power algorithm for a longer-lasting lifetime of smartphone location-based service (LBS) applications. The reason we focus on location awareness is that the LBS is being increasingly used by smartphone users and it is one of the killer applications of a smartphone [28]. Some examples of currently popular LBSs include social networking services (SNS), ubiquitous healthcare services, local traffic, and augmented reality. That is why the LBS is the application we are targeting to conduct our research on reducing power consumption in smartphones.

3.1 Analysis of power dissipation on smartphones

Unwanted power leakage occurs when it is not supposed to. Due to the inherent nature of a smartphone in which an operating system is working, power consumption continues even when the device is not being actively used. Smartphones play various roles that require many foreground and background jobs, whereas traditional feature phones only perform call processing. This results in severe power dissipation. The unwanted power leakage in smartphones is summarized as follows:

- Applications run as background jobs or services: user can install applications to run as background jobs periodically, such as with an Android service. For instance, once we set up Google mail sync, Android Google mail client will try to synchronize the content from Google mail and scheduler.
- Keeping Wi-Fi, Bluetooth and GPS enabled when we are not using the device: even though the applications that use those sensors are not working, just turning the sensors on consumes power. For example, the Wi-Fi protocol periodically communicates with access point (AP) from the beacon signals. This results in power consumption even on upper layer of Wi-Fi falling into the idle stage.
- Display lightness/contrast: the main concern with regard to power dissipation is due to the LCD brightness. The backlight power minimization can effectively extend battery life for mobile handheld devices.

To analyze the power dissipation behavior of the three classifications above, preliminary experiments consist of two types of workloads: synthesized workload and real workload. The synthesized workload is done to check the power consumption of each device (or sensor). Real workloads are done to check the power dissipation when popular applications are running. We start measuring with a fully charged battery after charging during the same amount of time. The screens of the smartphones are configured to always remain on. The GPS invocation interval is set to 5 seconds. To measure instantaneous battery levels of the phone over several hours, we make use of our battery-level monitoring application while running the two types of workloads. The battery-status monitoring application is based on Android service that runs in the background of other current activities. All tests and evaluations were performed with our battery status monitoring application [19] on Samsung Galaxy S [6]. Note that we ran the experiment multiple times, and we always observed the same trends in battery-level drops across all runs. The following figures display the preliminary experimental result with synthesized workload.

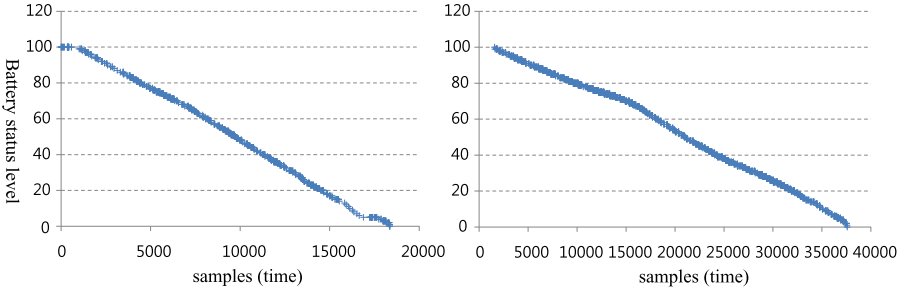

Fig. 1 Battery level logging when brightness is high and low

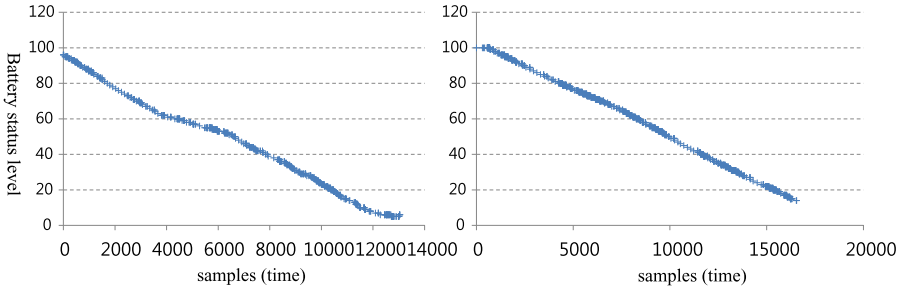

Fig. 2 Battery level logging when GPS and Bluetooth are enabled

We first assess the impact of using power-intensive LCD brightness on smartphones. Figure 1 shows the battery level of the phone during the run. When the brightness of LCD backlight is high, the battery level drops to 65% within two hours, whereas the battery level at low brightness drops only up to 90%. The battery lifetime of low brightness is almost twice as long as the lifetime of high brightness. Next, we check the impact of using a GPS (Global Positioning System) with smartphones. Figure 2 shows the battery consumption with using a GPS navigation application. When a GPS is enabled and used, the battery level drops to 63% within one hour. That means the smartphone battery level stays around 90% in an hour without a GPS, but the battery is exhausted up to 60% within an hour with the GPS enabled. This is because a GPS is one of the most power-consuming sensors or devices in a smartphone. For Bluetooth, the energy consumption is considerably less than that of a GPS. The right side of Fig. 2 plots the battery level logging on smartphones.

Figure 3 shows our experimental configuration: (a) power supply, (b) digital multimeter (True RMS Multimeter), (c) smartphone (Samsung Galaxy S), (d) laptop computer. The rated input voltage/current range of the Samsung Galaxy S is 1500 mA at 3.7 V. Assuming that the voltage difference is stably supplied with 3.7 V without a drop of electric pressure, measuring only current change with digital multimeter is the same as checking power dissipation. After connecting test leads in serial with the smartphone being measured, we log the change of current flow with the laptop computer (d) that is connected by USB with digital multimeter. Table 1 shows the average power dissipation for various workloads of synthesized and real cases.

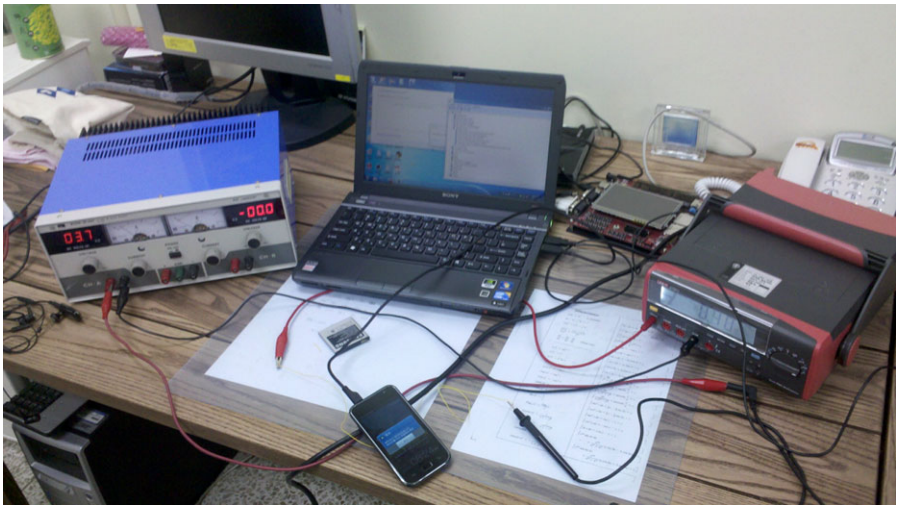

Fig. 3 Experimental configuration

Table 1 Workloads and average power dissipation

Synthesized workload	Test cases	Average
LCD backlight brightness	max/min	0.14∼0.51 mA
GPS	on/off	0.35 mA
Bluetooth	on/off	0.35 mA
Wi-Fi	on/off	0.47 mA
Real workload		
T-map (smartphone navigation application)		0.40 mA
Melon (music player application)	streaming/local	0.30∼0.51 mA
3G WCDMA call processing		0.25 mA

3.2 Energy-efficient location-based services

Typical smartphones are equipped with multiple types of sensors including Bluetooth, accelerometer, audio, camera, and GPS [26, 27]. The growth of sensors in smartphones drives applications in variety of fields, such as gaming, location awareness and augmented reality. However, due to the increase in this growing feature-sets and sensing capabilities, smartphones continue to suffer from battery life limitation. As was shown in Sect. 3.1, the aggressive use of LBS applications results in an exhausted battery within a few hours. Even though a power-intensive GPS hinders active utilization of location-based service, the GPS is still the core enabler of this type of service. In this research, we propose an energy-efficient framework for location-based service. The framework detects the smartphones' mobility state by using less-power-intensive sensors and eliminates unnecessary invocation of location sensing.

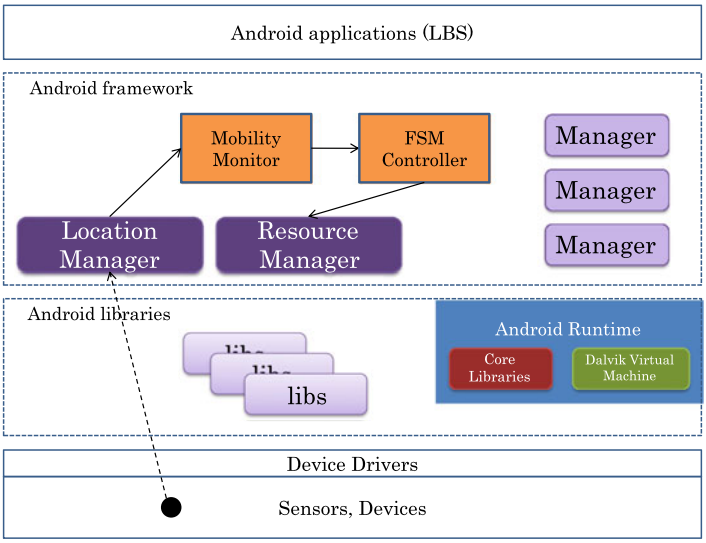

Fig. 4 Energy-efficient mobile-cloud computing framework architecture for PUC

This is because continuous location sensing may not be needed when the smartphone is in static state, such as being put on a table in the office. It is desirable to substitute the sensing operation with a more energy-efficient method. Actually, this framework reduces the power dissipation of a smartphone's location service by substituting less-power-intensive sensors for a long-lasting lifetime. When an accelerometer is combined with a GPS, we can save power in LBS applications. The accelerometer is mainly used to measure acceleration in perpendicular axes. It can sense tilt, motion and shock vibration.

Figure 4 shows energy-efficient mobile-cloud computing framework for pervasive and ubiquitous computing. Even though the framework is based on Android platform, the concept of proposed architecture will be applicable to other mobile platforms. In general, the location manager of Android framework gets data from the accelerometer sensor. If there are changes of values from accelerometer, the location manager notifies Android applications by calling OnLocationChanged(). From the continuous sequence of accelerometer input values, it is possible to detect whether a user is moving or not. The mobility monitor intercepts the values of latitude and longitude from the Location Manager. Then, the mobility monitor recognizes movement by comparing the acceleration value with thresholds (refer to Sect. 3.2.1).

Our design principle is to detect the movement of a smartphone with more energy-efficient sensors, allowing GPS hardware to sleep between successive location-updates. It depends on user movement which is detected by an accelerometer. When we move, gravitational acceleration changes because acceleration is an increase in speed or velocity. The accelerometer leverages power reduction techniques by telling a device to go into its low-power mode when that device is determined to be inactive based on the absence of movement or vibration. However, simply forcing applications to request GPS update less frequently is not the solution.

Table 2 Threshold settings in this experiment

Threshold	Explanation
TH_ACTIVITY	A threshold value for movement checking whether user is active or inactive. We assume user movement when acceleration is greater than the value TH_ACTIVITY. For this purpose, we set the threshold value to $0.5g$ in this experiment. Accelerometer provides value by three axes: x, y, and z. We make use of the sum of the three values of each axis. We assume the device is inactive when the sum of accelerations of less than the value TH_INACTIVITY is experienced for longer than the time specified by TH_SIGDUR.
TH_SIGDUR	The activity duration between two acceleration events that are greater than the value of TH_ACTIVITY and at the same time shorter than the time specified in the TH_SIGDUR value. We set this threshold as 1.5 seconds in this experiment.
TH_DIFF	The threshold for checking the acceleration difference between stable status and initial status. The TH_DIFF is set to $0.7g$ in this experiment.

To realize the design principles, we provide two components: a movement recognition strategy and a power management finite state machine (FSM). Movement recognition is to check whether a user moves. Many threshold values are necessary to implement movement recognition. Power management is a resource scheduling policy based on the FSM in which two state transition triggers exist. One is user movement and the other is timer expiration. When a user moves or a timer expires, our framework goes back to normal state or starts low-power mode, respectively. To this end, we provide a power (LP) timer that provides a method for the device to automatically enter the low-power mode from either the Active or the Idle state following a programmed period of inactivity. We make use of two timers, namely the LP1 timer and the LP2 timer.

3.2.1 Movement recognition strategy

Movement recognition takes into account the changes in acceleration that occur when a smartphone is moved. It is based on the principle of detecting changes in motion and position of the smartphone. However, it is quite challenging because it has to track acceleration changes and perform continuously algorithmic analysis to detect movement. The core element of movement recognition is an effective, reliable detection principle and an algorithm to determine the existence of user movement.

While the vector sum of accelerometer input is generally around $1g$ under normal conditions, the user movement causes vector sum to fluctuate over $1g$. For this activity detection, the acceleration value at the beginning of activity detection is taken as a reference value. At the start of LBS applications, the reference value is used for compensation of our movement recognition strategy. Then, we need to predict whether an application will be using the GPS or not. In case of Android applications, all applications have a configuration file, known as AndroidManifest.xml. The XML file is used to represent which devices and sensors are required for the application. In Table 3 we see the following statements in the XML when the application requires the GPS, 3G cell tower triangulation, Internet access, and camera device.

Table 3 A part of AndroidManifest.xml

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.LOCATION"/>
```

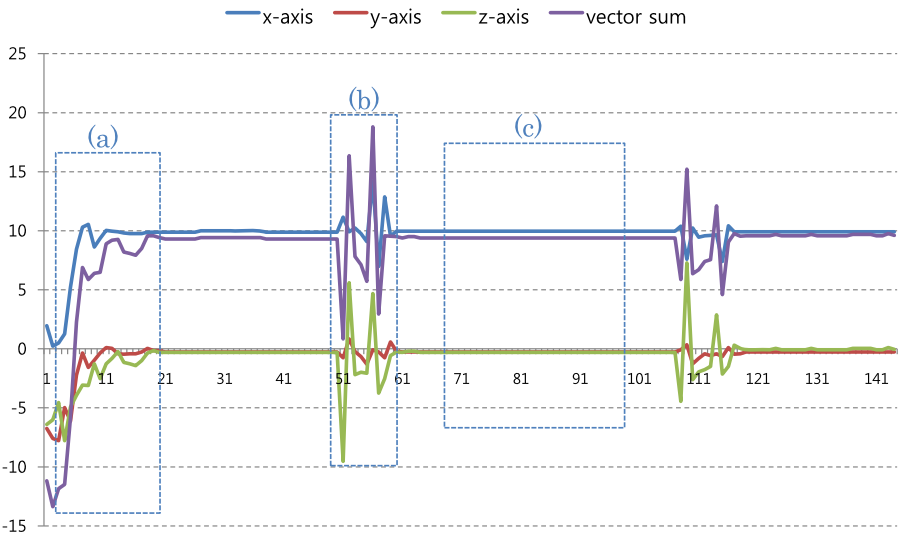

Fig. 5 Acceleration patterns on the event of movement

New samples of acceleration are then compared to the reference value. If the magnitude of the difference exceeds TH_ACTIVITY, the device will indicate that user movement has taken place.

Additional information for movement detection in our framework is the difference between acceleration and orientation. After movement, the device will be in a different orientation than before, so the static acceleration in three axes will be different from the initial status before the movement. Therefore, if the acceleration difference between stable status and initial status exceeds the TH_DIFF threshold, a valid movement is detected.

The accelerations during movement are different in terms of patterns or behaviors in the event. Figure 5 shows the acceleration changes during the movement of a car or a pedestrian. By comparing with normal states, we can see three critical different characteristics of the movement that can serve as the criteria for the event detection. They are indicated by the dashed boxes in Fig. 5 and explained in detail as follows: (a) INITIAL_STATE, (b) EVENT, (c) STEADY_STATE. Since INITIAL_STATE is the unstable and unreliable state that is not properly initialized, we ignore this state. At the first period of STEADY_STATE, we take acceleration inputs as the reference values. We use this reference value to recognize user movement with the thresholds

in Table 2. At the period of EVENT, the smartphone body will impact from movement: the acceleration curve shows this as a large shock. This shock is detected by the following condition checking: the activity duration between two acceleration events that are greater than the value of TH_ACTIVITY and at the same time shorter than the time specified in the THRESH_SIGDUR value. STEADY_STATE always occurs when movement is not significant. Mobility manager in Fig. 4 classifies this state as movement. It becomes more significant on movement, and the vector sum of accelerations satisfies the following condition when the vector sum is the total summation of the accelerations of x, y, and z.

$$|\text{vector sum}| < 0.5g$$

The reason why we are using the sum of vectors x, y, and z is that the movement is not in one direction. Surely when this strategy runs on navigation, the movement will not usually be only back and forth. However, the movement direction targeted for pedestrians is not one way. Therefore, we need to use the summation, rather than only one element. This technique can reliably detect the start and duration of significant user activity.

Moreover, there is another problem which is that the accelerometer readings are not stable. Typically, there are fluctuations of about 2–5% even when the device is resting on a table. When we look at raw data from the accelerometer sensor, it is surely going to fluctuate. To resolve this problem, we make use of a filter, the trailing weighted average of current and past rates that can be used to determine movement. In our experiment, the currently up-to-date value is weighted at 95% in computing the average, and the past data are weighted at 5%. When movement is detected by a larger change than the threshold, a signaling event is presented to the FSM as one of the inputs for power management.

3.2.2 Power management finite state machine (FSM)

The high accuracy of location awareness may increase with increasing power management levels. This means that device's power consumption may increase with increasing accuracy levels of location-based services due to the power-accuracy trade-off. Today's smartphones achieve high accuracy at an acceptable level of power dissipation. In terms of power accuracy trade-off, the GPS is particularly important because it is the core enabler of location-based services, but aggressive use of the GPS can severely increase power consumption. In this research, we propose an efficient power management policy for location-based services. The policy is implemented by the FSM as shown in Fig. 6. The FSM controls the state transition depending on its input and current state. The states in our power management FSM include the GPS sensor activity status (Active, Idle, LP1, LP2). The FSM state transition is triggered by movement detection or timer expiration.

A device may implement one power management method for two or more contiguous power management levels. Actually, we provide three step power reduction states as in Table 4, in addition to the "ACTIVE" and "IDLE" states. These states are "LP1," "LP2" and "LP2 with System Power Management (PM)" which differ

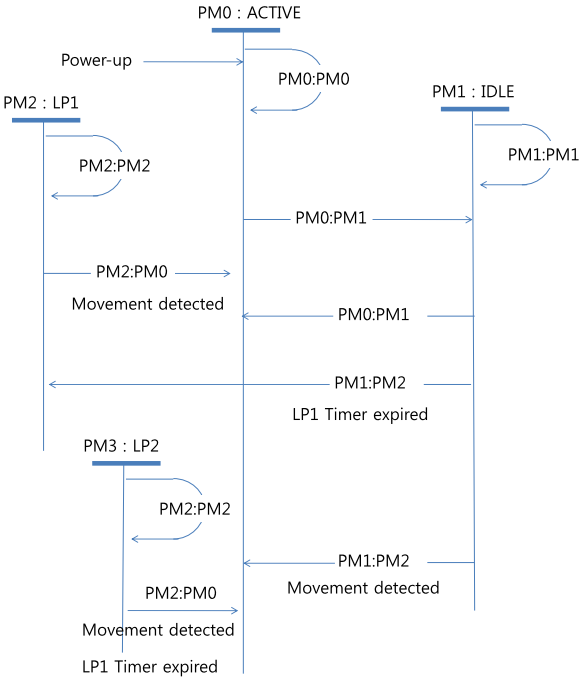

Fig. 6 Power management FSM

with regard to the level of power reduction and the return latency. LP1 has maximum return latency of 10 microseconds, while "LP2" has a maximum return latency of 10 milliseconds. Therefore, we can realize greater power savings in "LP2" and "LP2 with System PM" with its longer specified return latency. "LP1," therefore, is designed to use a relatively simple power reduction technique with minimal impact on performance. "LP2" is designed to be used only when the device is expected to be idle for an extended period of time. Transition of the device power mode directly from "ACTIVE" to "LP2" without passing "LP1" is not possible.

Even though our solution can potentially be applied to any mobile platform that deploys location-based services, we present the architecture specifically on an Android OS for ease of presentation and concrete implementation. Such a selection is also justified because of the Android's open nature and increasing popularity. Note that the architecture and design principles can also be implemented on other platforms, including Symbian and Windows Mobile. Our framework is realized as a middleware solution, residing between applications and underlying Linux kernels. Specifically, the Android platform includes an application framework that packages many useful classes in Java. The solution is implemented inside the Android application framework by modifying existing classes as well as creating new classes.

We prototype the proposed solution on a Samsung Galaxy S Android phone equipped with OS version 2.1 (Eclair). All four design principles are implemented in Java inside the Android framework. The prototype contains a graphic user interface (GUI) that allows a user to enable, disable and finely configure the prototype. With current Android APIs, the GPS is invoked through a major function call, requestLocationUpdates(), which takes at least four input parameters: LocationProvider, which

Table 4 Each state and explanation in power management FSM

State	Explanation
ACTIVE	This state is entered when acceleration greater than the threshold value THRESH_MOVEMENT is experienced. At this state, the device is fully powered-up and all sensors are actively running. This state shall be entered when the device detects movement while in Idle or Standby mode. This state shall also be entered when the device is powered-up.
IDLE	"IDLE" is the state when acceleration of less than the value stored in the THRESH_INACT register is experienced for longer than the time specified in the TIME_INACT register. The device is able to response instantly to external input in the "IDLE" state. No movement is detected at this time. The device stays in the "IDLE" state all the time except when actively working. "IDLE" state also sets the "LP1" and "LP2" timer count.
LP1	"LP1" saves about half as much power as LP2, but with significantly less impact on performance. LP1 has a maximum return latency of 10 microseconds.
LP2	"LP2" is used if device will be idle for a significant amount of time, but you are willing to sacrifice some performance for power savings. It takes time to respond to your movement. "LP2" has a maximum return latency of 10 milliseconds.
LP2 with System PM	"LP2 with System Power Management (PM)" state is still the "LP2" state in our framework, but it is the lowest-power state because, at this state, the system power management goes to minimum power state such as sleep, which is standby in Linux Operating Systems. Thus, it is the low-power state of both our framework and a Linux OS at the same time.

reports frequencies in terms of time and distance, and PendingIntent or LocationListener. Our prototype mainly captures this function call and embeds intelligence inside the function as well as other relevant functions.

In addition, our architecture provides an optional feature, known as Power-up in LP2 (PUILP2). The PUILP2 prevents the smartphone from automatically enabled on device boots up. Therefore, activity for accessing of GPS sensors starts only when the location-based service launched, to conserve electric power.

4 Mobile cloud convergence

Mobile cloud convergence is a paradigm shift in field of mobile and parallel computing. In the next few years, we can expect a major shift from traditional mobile application technology to mobile cloud computing. It improves application performance and efficiency by offloading complex and time-consuming tasks onto powerful computing platforms. By running only simple tasks on mobile devices, we can achieve a longer battery lifetime and a greater processing efficiency. This offloading with the use of parallelism is not only faster, but it can also be used to solve problems related to large data sets of non-local resources. With a set of computers connected on a network, there is a vast pool of CPUs and resources, and you have the ability to access files on a cloud. In this paper, we propose a novel approach that realizes the mobile cloud convergence in transparent and platform-independent way. Users need not know how their jobs are actually executed in distributed environment and users need not take into account whether their mobile platforms are iPhone or Android.

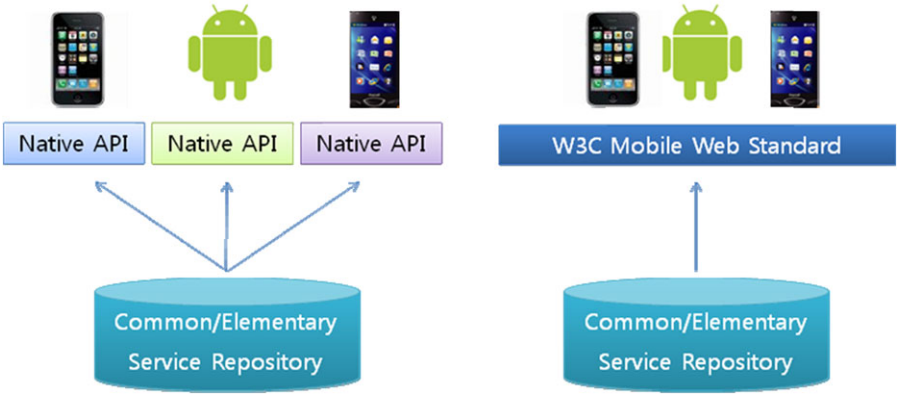

Fig. 7 Smartphone application development environment

This is because the core technology of our framework is based on web service and SOAP protocol through HTTP 80 port. We are targeting on OS-independent smartphone application development platform. The OS-independent platform means that the structure is not depending on smartphone platform. It consists of web service and mobile web (HTML5 like) device API standard. By running complex task on cloud as a web service it is possible to reduce computation time and battery power. Likewise, complex business logics and computations will be offloaded by cloud computing platforms.

The left side of Fig. 7 shows current smartphone application development platform including Android, iPhone, and Windows Mobile. Each platform has its own applications development environment and they are not compatible to others. If you want to use an iPhone application, you must have iPhone. Similarly, if you want to enjoy some Blackberry applications you have to have Blackberry. On the other hand, the right side of Fig. 7 represents platform-independent smartphone application development environment. With this mobile cloud computing framework you will be able to enjoy all such applications only if you can access web through your cell phone. This framework provides a commonly accessible layer which is platform independent, for example W3C mobile web standard. Likewise, our proposed approach makes use of web service architecture through the W3C mobile web standard layer.

However, only providing facilities such as vast pool of servers is not sufficient for mobile cloud convergence. For offloading mobile computation by cloud service, framework support is necessary. Especially, the support has to be of a service-oriented type. A simple remote procedure call (RPC) or an inter-process communication (IPC) in cloud side is dependent on a certain platform or target, such as Android or iPhone. So, this is not the proper solution of general purpose offloading framework.

In terms of platform independence, Android IPC is better than the RPC or IPC since it utilizes a platform-independent interface definition language (IDL). Actually, Android IPC runs as an Android service which runs at background while other activities (applications) are running at foreground. The service can be utilized between multiple activities. Once an Android activity is bound to currently executing service,

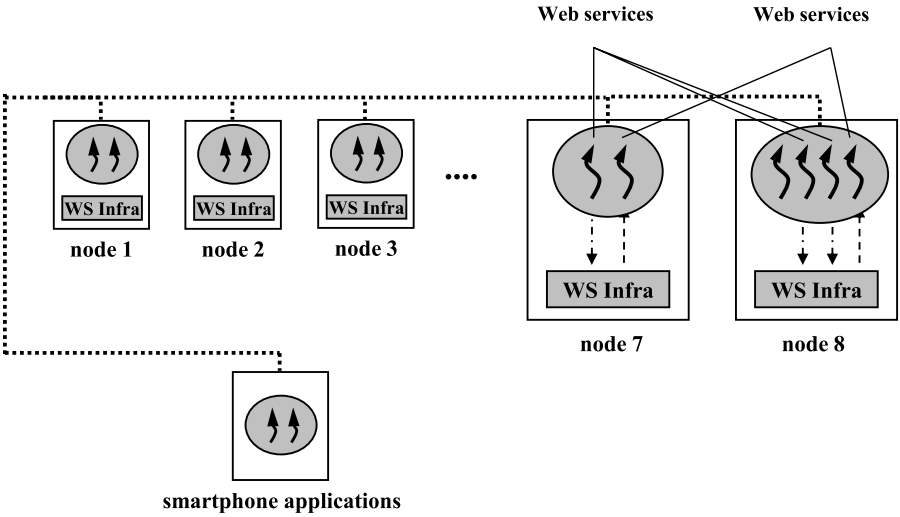

Fig. 8 Mobile cloud convergence framework with web service

it will communicate with the service using a predefined interface by a stub/proxy pair. The service interfaces are defined in an interface definition language, known as AIDL. But, the Android IPC still has a drawback in that it is only able to communicate within a local node. The Android IPC still has a limitation because the communication technique is localized within the local platform. The Android IPC is not applicable when we are trying to communicate with other computers or devices. Thus, if we want to communicate with other sides, it is better to use network socket instead of Android IPC. Therefore, our smartphone application development environment does not depend on any native platform. This is to provide environment of platform-independent smartphone application development without concern of the OS.

In addition to the platform independence, one of the important things in area of smartphone application development is the fast implementation or fast delivery (rapid application development: RAD).

If an idea is realized as a smartphone application by a developer, the first development is being the standard in the area. For this fast development and quick deployment, development method by building up with commodity components is more recommendable than implementing from scratch. Through reusable and composable elementary components, we can easily make new services and applications very quickly. In this paper, our mobile cloud architecture is based on web service and we provide common service and composable elements as a sort of web service. Figure 8 shows job distribution strategy of PI value computation on our mobile cloud convergence framework with web service. By this offloading framework, computation oriented parts run on a remote cloud node as a web service and the rest of the parts run on mobile devices. As a result, mobile cloud convergence framework leads to performance improvement and longer battery lifetime of mobile devices.

Figure 9 depicts the concept of the service oriented smartphone application development framework. The web service oriented infrastructure is realized by SOAP

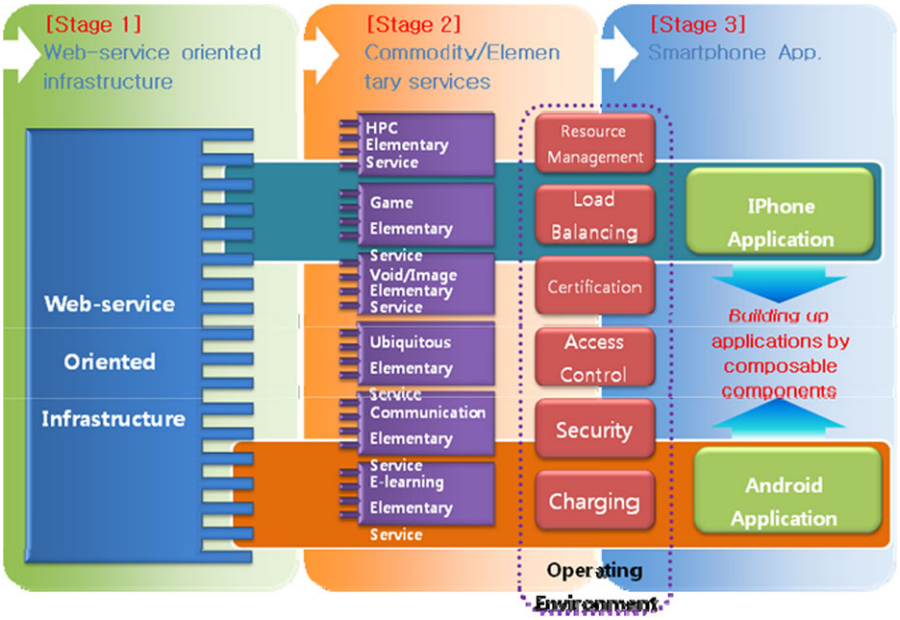

Fig. 9 The conceptual view of web service oriented smartphone application development framework

with Attachments API for JAVA (SAAJ). This framework provides 6 key administration policies and services: resource management, load balancing, certification, access control, security, and charging. As shown in the stage 2 of Fig. 9, we can establish a service repository consisting of elementary/commodity services. All these commodity/elementary services are implemented and deployed by a type of web service. Thus, this is totally platform-independent and fast deliverable for smartphone application development. For the convenience of smartphone application development, we expect that the following common/elementary services are necessary: augmented reality, HPC, voice/image processing, ubiquitous computing, E-learning, and so on. However, due to the limit of time and budget, we currently provide only PI value computation web service as a prototype. In Fig. 9, applications on mobile devices need to locate their wanted service. For this purpose, web service architecture comes with Universal Description, Discovery and Integration (UDDI). UDDI is a platform-independent XML based registry worldwide to list themselves on the Internet and a mechanism to register and locate web service application. Our framework also makes use of UDDI to locate the proper user wanted web service.

4.1 Task parallelization of π calculation

In this section, we show a development procedure of the cloud-based applications on a mobile platform, especially π calculation. The first step in building the mobile cloud converged application is to identify sets of tasks that can run concurrently and/or partitions of data that can be processed concurrently. The second step is to eliminate dependency, if any exists, between every computational phases in the algorithm. The dependency limit of the degree of parallelism results in performance degradation.

π is a mathematical constant whose value is the ratio of any Euclidean plane circle's circumference to its diameter; it is the same value as the ratio of a circle's area to the square of its radius. Many formulas from mathematics, science, and engineering involve π, which makes it one of the most important mathematical constants. The simplest method to calculate π is circumference divided by diameter. However, it is difficult to get the exact circumference using this simple method. As a result, there are other formulas to calculate π. These include series, products, geometric constructions, limits, special values, and π iterations. To calculate π through mobile-cloud convergence, we first need to convert the algorithm into a parallelized version. We present a π calculation with infinite series that puts forth a parallelization method for ease of application on the mobile cloud convergence. First, our π calculation involves the following Maclaurin series of the inverse tangent function (1):

$$\tan^{-1}(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{2n+1} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \frac{x^{11}}{11} \cdots \tag{1}$$

The case of the Taylor series for $c = 0$ is the Maclaurin series. Therefore, (3) is found by taking (2) and evaluating for the special case $c = 0$. To calculate π, we compute the Maclaurin series generated by $f(x) = \tan^{-1}(x)$ as shown above.

$$\tan^{-1}(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots + (-1)^n \frac{x^{2n+1}}{2n+1} + \cdots$$

A function to compute this based on the above form cannot be parallelized because each computed value is dependent on previously computed values. For example, to calculate $\tan^{-1} = \frac{\pi}{4}$, we need to compute $\tan(1) = 1 - \frac{1^3}{3} + \frac{1^5}{5} - \frac{1^7}{7} + \cdots$. We have to compute the partial result of $1 - \frac{1^3}{3}$ to add $\frac{1^5}{5}$ to the partial sum. To offload the computation-bound part to the cloud, an independent form of this equation should be provided.

To this end, we convert the equation into an integral form that is suitable for mobile cloud convergence. Take the derivative of the above equation with respect to x, and change the variable x to t, for the sake of convenience.

$$\frac{d}{dt}\tan^{-1}(t) = 1 - t^2 + t^4 - xt^6 + \cdots + (-1)^n t^{2n} + \cdots$$

Substituting $\frac{1}{1+t^2} = 1 - t^2 + t^4 - xt^6 + \cdots + (-1)^n t^{2n} + \cdots$ into this formula we get the following:

$$\frac{d}{dt}\tan^{-1}(t) - \frac{1}{1+t^2}$$

Integrating this equation for the interval a to b yields the integral form of $\tan^{-1}(t)$:

$$\tan^{-1}(t) = \int_a^b \frac{1}{1+t^2} \quad (\forall a, b \in R)$$

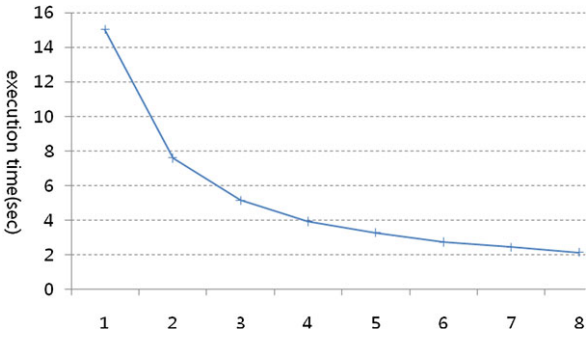

Fig. 10 Mobile cloud performance as # of cloud nodes

By substituting $\frac{\pi}{4} = \tan^{-1}(t)$ into this formula we get parallelized form that is executable on cloud computing platform:

$$\pi = 4\tan^{-1}(t) = \int_a^b \frac{4}{1+t^2} \quad (\forall a, b \in R)$$

We approximately get the π value by integrating this equation for the interval $-\frac{1}{2}$ to $\frac{1}{2}$. Unlike an infinite series representation, the integral form is fully parallelizable and it is easy to divide the problem into chunks/parts of work. We distribute and map these tasks onto multiple clouding nodes. However, this equation cannot be executed on cloud computing which is highly parallelized and distributed in a computing environment. This is an example of task parallelization and partitioning, and it can be run on a mobile cloud convergence platform.

We constructed our system with 8 nodes from cloud service, of Core2 Duo 2 GHz machines each with 2 GB RAM. The machines are connected by 1 Gbps Ethernet. Figure 10 shows an overview of our mobile cloud convergence framework, designed in 3 stages: the first one is web service oriented infrastructure, the second one is commodity/elementary web services, and the third one is operating environment.

As the number of nodes increases, the total execution time for computing π value decreases. The following Table 5 shows the execution time for each cloud node. Since our π value calculation algorithm distributes the same amounts of data to all participants, the execution times in a row are almost the same. And the final execution time contains more time such as communication overhead, fork-join overhead, processing overhead on mobile devices. A row in Table 5 represents the computation participants; for example, line 4 means that our π value computation algorithm distributes the same amount of data to all participants. The plotted graph value in Fig. 8 becomes the sum of maximum value in Table 5 with TCP communication overhead.

5 Seamless connection handover on cloud computing

At present, dual-mode 3G and Wi-Fi smartphones are popular. The requirements of data communication using cellular network are significantly growing. So, mobile operators try to set up Wi-Fi APs on streets in order to offload traffic to Wi-Fi systems to reduce cellular traffic congestion, as shown in Fig. 11. In this environment, we need

Table 5 π value calculation result (execution time) for each cloud node

	node 1	node 2	node 3	node 4	node 5	node 6	node 7	node 8
1	14.703							
2	7.343	7.344						
3	4.906	4.89	4.907					
4	3.688	3.672	3.672	3.688				
5	2.953	2.937	2.954	2.938	2.937			
6	2.453	2.453	2.453	2.453	2.454	2.453		
7	2.11	2.11	2.093	2.109	2.109	2.094	2.094	
8	1.843	1.844	1.843	1.829	1.828	1.844	1.828	1.828

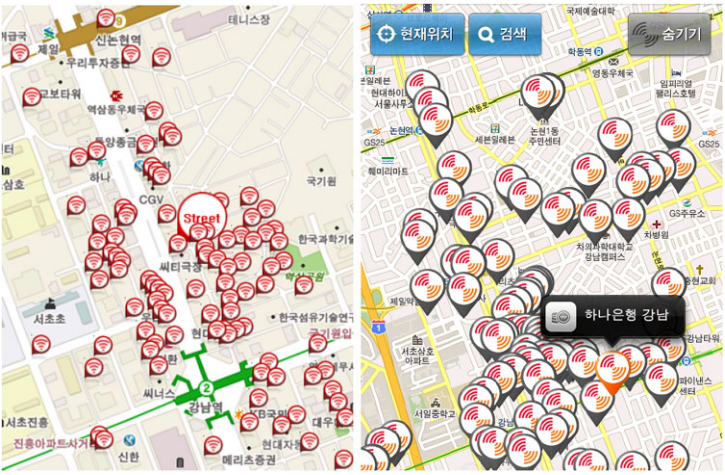

Fig. 11 Currently installed Wi-Fi hotspots in KangNam-gu, Seoul

to provide the seamless connection handover between different access networks. For convenient establishment on site, our approach is based on the end-to-end architecture between server and smartphone that is not depending on the internal architecture of cellular networks.

With the increasing deployment of mobile computing, the interest for connection handover is again on the rise. The connection handover is the act of transferring a process between two machines during its execution. There are two most challenging issues. Recently, the emergence of network-based distributed and mobile computing environments has increased the necessity of connection handover technology, especially in the industrial and research field. The research and development for connection handover mechanism based on the Android operating system is very important and inevitable. This study aims at development of seamless connection handover. Nowadays, practical uses of connection handover for the improvement of working environment have increased rapidly. It is applicable for seamless services of mobile devices, work space preservation in PUC.

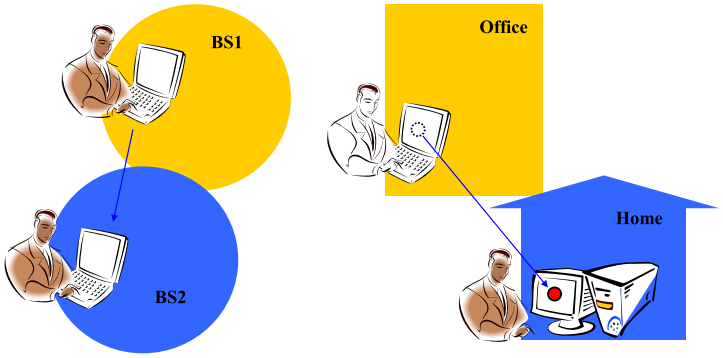

Fig. 12 The necessity of connection handover

This research is focusing on development of seamless connection handover for Wi-Fi/3G wireless Internet. These days, mobile devices do not support the seamless handover between 3G and Wi-Fi. Therefore, when we move from an AP (access point) of Wi-Fi to another AP, or when we move from Wi-Fi network to 3G-based cellular network, currently executing applications are terminated and return errors as shown in Fig. 12. In order to resolve the problems, we provide a novel method to support seamless connection handover technique. Our approach does not affect the network architecture that is already commercially established, since our technique is fully end-to-end approach. This is fairly creative and challenging research topic that has not been taken into account for any research groups in the world. So, we develop and research on the following issues. Simultaneous horizontal/vertical handover: when smartphones are moving among Wi-Fi hotspots, or moving between Wi-Fi and 3G, connection termination occurs. We provide the following to avoid the termination in these situations. (1) Architecture with no home agent: The reason why the Mobile IP protocol is not broadly used these days is the "home agent." It is a specialized router that controls the movement of mobile nodes and provides tunneling service between servers and the mobile devices. (2) End-to-end approach: our technique is fully end-to-end approach. This is fairly creative and challenging research topic. This approach does not affect the network architecture which is already commercially established.

5.1 Broken pipe problems

On connection handover, the major problem with a socket is the messages in transit. Simply checkpointing the process state, regardless of the network state, results in an inconsistent application state after restoration. Moreover, broken pipe problems happen when the sender writes a packet, even though the receiver process had been interrupted and the socket was closed, as shown in Fig. 13(a). In order to resolve this problem without capturing network state, we make use of communication channel clearing before handover.

Clearing communication channel is achieved by zero window advertisement. The zero window advertisement (ZWA) interrupts the data transmission by adjusting the window of sender process to 0 as in Fig. 13(b). Since the ZWA is a network level

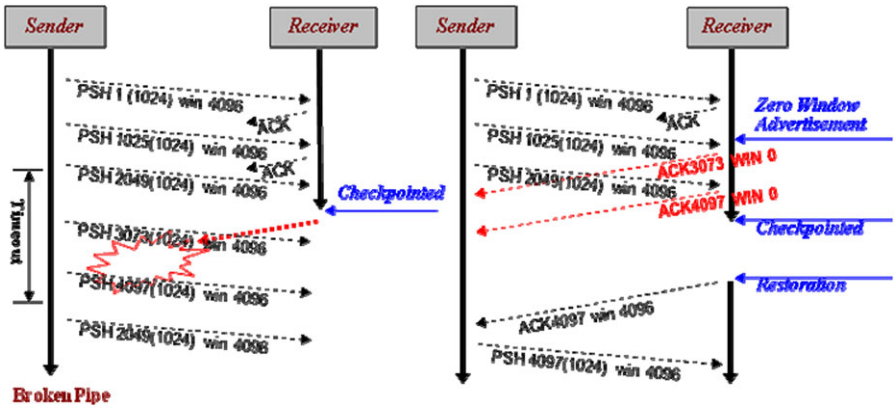

Fig. 13 Broken pipe problem and the solution approach

approach, the peer application can try to write data continuously without knowing that the send window is closed. Thus, we suspend the application process right after the ZWA. As a result, at the moment of connection handover, we can guarantee that there are no transit data on the communication channel. Using this clearing communication channel, the socket of the process is migrated as follows:

(1) The socket as sender
It is quite simple to move the socket of a migrated process when it acts as a sender. The socket does not merely write data. That is, it just does the handover without any special preprocessing such as zero window advertisement or application process suspending.
(2) The socket as receiver
When the socket of a migrated process is the receiver, as in Fig. 13(b), the receiver advertises a zero window to the sender. Even if advertising zero window during the handover, there is no problem with receiving a packet in transit.
(3) The socket as both sender and receiver
This is the typical case on distributed systems in which processes communicate by sending and receiving data. In this case, we take the approach of both 1 and 2 on each node. Thus, it interrupts data sending of corresponding nodes by zero window advertisement.

The distributed system is composed of a collection of processes. A process can be viewed as consisting of a sequence of events. In the process, we do not need to know the elapsed time between two events. But, we have to know the order of two events. Thus, we assume that the events of a process form a sequence, where A occurs before B in this sequence if A happens before B. This is the definition of the "happened before" relation. The relationship is given with an arrow: $\rightarrow$. What happens before relation is a transitive relation, so if a $\rightarrow$ b and b $\rightarrow$ c, then a $\rightarrow$ c. Figure 14 shows the message exchange for the connection handover. In the message exchange procedure, we put some constraints on the order of messages. The constraints must be fulfilled to guarantee successful connection handover as follows:

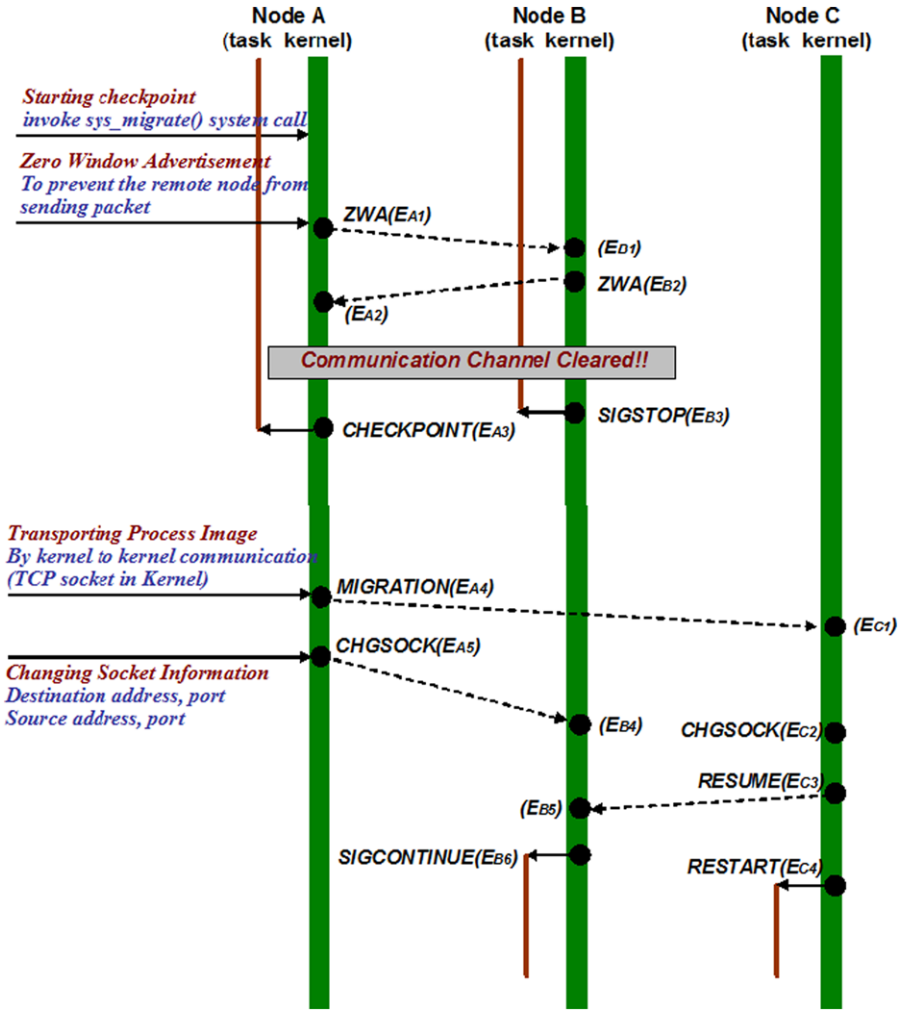

Fig. 14 Sequence of events during connection handover

(1) The event that node A advertises its window size as zero must happen before the event that a corresponding process on node B suspends its execution with SIGSTOP. This is because EA1 → EB1 and EB1 → EB3, from which, by transitivity property above, it also follows that EA1 → EB3.
(2) The event that node A transfers the saved process image to node C happens before the event which resumes the suspended process on node B. This is because EA4 → EC1, EC1 → EC3, EC3 → EB5, and EB5 → EB6, therefore EA4 → EB6 by transitivity.
(3) It makes no difference which of the two nodes suspends the process first since there are no packets in transmit between the processes of node A and node B by zero window advertisement. Hence, the order is off and does not matter.

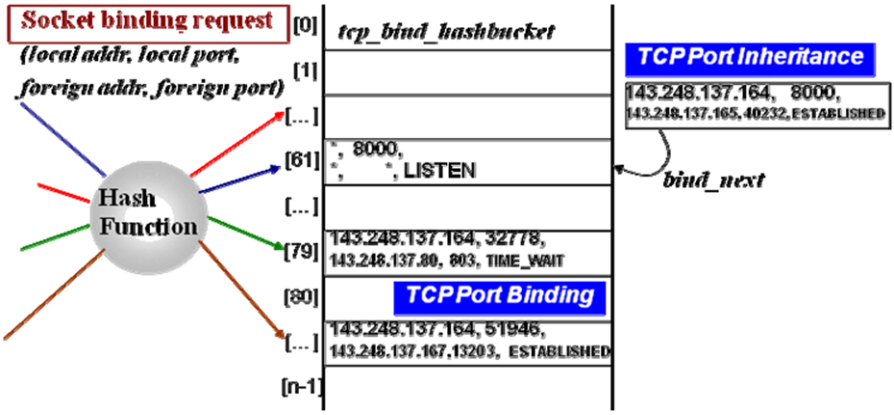

Fig. 15 The difference between TCP port binding and TCP port inheritance

(4) The event which resumes the suspended process on node B has nothing to do with the event which resumes the transferred process image on node C. This is because any process does not send data by zero window advertisement. Hence, the order is off and does not matter.
(5) There is also no relevance to the order of zero window advertisement. Though its socket is frozen by zero window advertisement, node A can receive data from the corresponding sender node B and vice versa. Therefore, the order is off and does not matter.

5.2 Binding errors on socket reconstruction

If a process operating on a server has a listener socket and an incoming connection request has been arrived from a client, the server creates a child socket and accepts the request. Previous researches did not offer server socket handover because of a serious bind error problem on socket reconstruction. The Linux kernel makes use of a hashtable, as in Fig. 15, to facilitate maintenance and lookups of socket objects. The information of source and destination of a socket object is used as the hash key. For a binding and listening socket, the source port is used as the hash key. For a connected socket, a tuple of saddr, sports, daddr, and dport is used to create the hash key. When socket binding request comes to kernel, the hash function tcp bhashfn transforms the hash key into a hashbucket address in the hashtable tcp bhash. If the socket information is identical to an already binded socket (in this event the same source port being used by both a listening and child socket), then a bind error occurs.

This is because they get the same hashbucket address from the hash function. In order to resolve this problem, we make use of TCP port inheritance to reuse the same source port from the parent socket. Thus, more than two socket objects are chained on the same hashbucket and share the same port. In Fig. 15, both parent and child sockets use the same source port of 8000. In order to reconstruct a socket during handover, the parent socket must be created first and enter the TCP LISTEN state. Then the child socket is created and inherits the parent's port. The child socket gets the address of

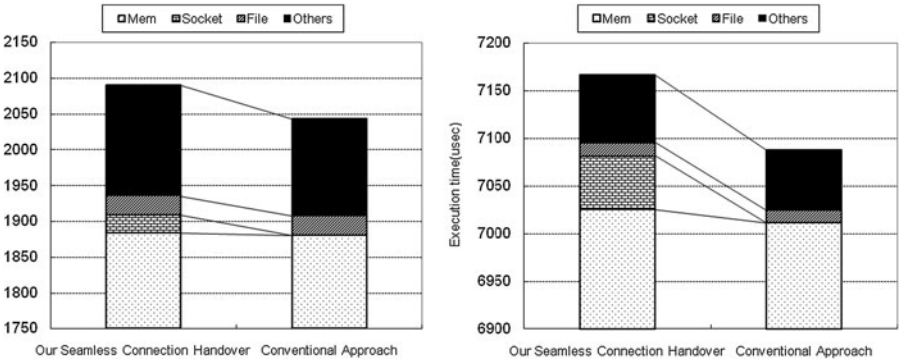

Fig. 16 Connection handover overhead

the tb → owner stored in prev of the parent socket. Then it saves the address of the parent socket in the "bind next" field. Therefore, it is possible for an established child socket object to put itself into the hashtable structure without binding operation.

An evaluation of the implementation of our proposed framework shows that it saves a significant amount of energy in the process of location sensing. To measure the cost of connection handover, we performed the process of checkpointing and restarting, which exchanges "ping-pong" messages repetitively. Figure 16 displays the connection handover overhead. In the figure, the "File" represents the overhead to save the process' file descriptors, and "Others" represents the remaining time consumed by other tasks, such as initializing global variables or running a loop in control logic. According to the figure, the vast majority of time is spent doing a memory dump. In Fig. 16, the y-axis does not start from 0 so that the reader can easily see the difference.

The time to save and restore socket structure takes only 0.7% of the checkpoint time and 1.1% of the restart time, respectively. This implies that the overhead due to socket structure handover is almost negligible when compared with the time needed for the basic process checkpointing and restarting. The zero window advertisement stops transmission for a limited period of time, resulting in decreased TCP throughput. Thus, we need to evaluate the effect of zero window advertising on TCP throughput. Figure 17 displays the expected amount of blocked data as a function of window size.

In Fig. 17, the blocked data size on the y-axis is obtained by multiplying the TCP sending rate by the blocking time due to the handover (Tm). The send rate of a TCP packet can be approximated by window size (in bytes) divided by round trip time (RTT) [24]. The average blocking times are about 7, 8 and 9 ms, which are derived from the experimental result of the handover overhead. Further, we assume that the RTT is 100 ms and the window size changes from 1 to 64 KB. As observed in the figure, when the window size is lower than 32 KB, the totally blocked data size is lower than 3 KB. If the packet size is 1 KB, fewer than three packets are blocked. Therefore, we can infer that the TCP throughput will not be seriously degraded if the handover blocking time is reasonably small.

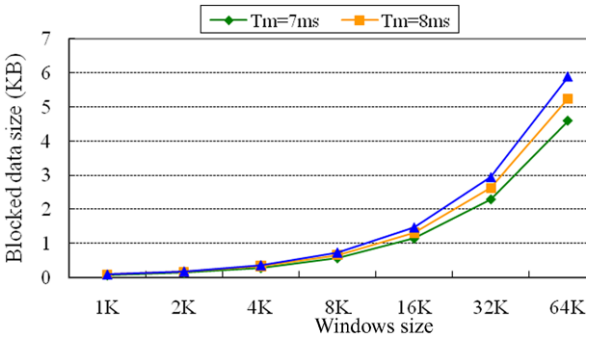

Fig. 17 Connection handover effect on TCP throughput

6 Concluding remarks

The results we have presented so far suggest that significant energy-efficiency, development convenience, rapid development/deployment can be obtained with a collection of techniques in this research. We consider the problem of power dissipation, development/deployment convenient, and connection handoff. We first identify and analyze the power dissipation in of variety configurations of smartphone usage. After that, we propose energy-efficient location-based service (LBS) and mobile cloud convergence. This framework reduces the power dissipation of LBSs by substituting power-intensive sensors with less-power-intensive sensors, when the smartphone is in a static state, for example, when lying idle on a table in the office. The core elements of this framework are a movement-detection algorithm to determine user movement, and a power reduction strategy controlled by an FSM. The FSM state transition is triggered by movement detection or timer expiration. We also propose a solution for the two major problems faced when attempting to realize seamless connection handover. The broken-pipe problems are resolved by zero window advertisement. The bind error during socket reconstruction is eliminated by TCP port inheritance. This is a highly transparent approach, in that it does not introduce additional messages for channel clearing and does not make any modification to the TCP protocol stack. For convenient on-site establishment, our connection handover approach is based on the end-to-end architecture between a server and a smartphone, which is independent of the internal architecture of current 3G cellular networks.

Acknowledgement This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0027161 and 2011-0003896). We would like to thank Prof. Seungho Lim of Hankuk University of Foreign Study for technical improvement.

References

1. WiMAX (2008) Through 2013—An Evolution to 802.16m, http://www.researchandmarkets.com/reports/577457/wimax_2008_through_2013_an_evolution_to_802.pdf
2. http://www.gartner.com/it/page.jsp?id=1543014
3. Open handset alliance. http://www.openhandsetalliance.com/
4. http://en.wikipedia.org/wiki/Cloud_computing

5. Vance J (2011) Mobile cloud computing: 5 key trends, http://itmanagement.earthweb.com/netsys/article.php/3922856/Mobile-Cloud-Computing-5-Key-Trends.htm, January 31, 2011
6. SAMSUNG Galaxy S, http://www.samsung.com/global/microsite/galaxys/
7. Facebook. http://www.facebook.com/
8. Myspace. http://www.myspace.com/
9. Twitter. http://www.twitter.com/
10. Azizyan M, Choudhury RR (2009) Surroundsense mobile phone localization using ambient sound and light. SIGMOBILE Mob Comput Commun Rev 13(1):69–72
11. Brezmes T, Gorricho J-L, Cotrina J (2009) Activity recognition from accelerometer data on a mobile phone. In: Proceedings of IWANN '09, Salamanca, Spain
12. Györbíró N, Fábián A, Hományi G (2009) An activity recognition system for mobile phones. Mob Netw Appl 14(1):82–91
13. Lester J, Choudhury T, Borriello G, Consolvo S, Landay J, Everitt K, Smith I (2005) Sensing and modeling activities to support physical fitness. In: Proceedings of UbiComp '05, Tokyo, Japan
14. Campbell AT, Eisenman SB, Fodor K, Lane ND, Lu H, Miluzzo E, Musolesi M, Peterson RA, Zheng X (2008) Transforming the social networking experience with sensing presence from mobile phones. In: Proceedings of ACM SenSys '08, Raleigh, NC, USA
15. Gellersen HW, Schmidt A, Beigl M (2002) Multi-sensor context-awareness in mobile devices and smart artifacts. Mob Netw Appl 7(5):341–351
16. Kang S, Lee J, Jang H, Lee H, Lee Y, Park S, Park T, Song J (2008) Seemon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments. In: Proceedings of ACM MobiSys '08, Breckenridge, CO, USA
17. Miluzzo E, Lane ND, Fodor K, Peterson R, Lu H, Musolesi M, Eisenman SB, Zheng X, Campbell AT (2008) Sensing meets mobile social networks: The design, implementation and evaluation of the CenceMe application. In: Proceedings of ACM SenSys '08 Raleigh, NC, USA
18. Wang Y, Lin J, Annavaram M, Jacobson QA, Hong J, Krishnamachari B, Sadeh N (2009) A framework of energy efficient mobile sensing for automatic user state recognition. In: Proceedings of ACM MobiSys '09, Kraków, Poland
19. Choi M Monitoring application: Battery status monitoring application (available at http://sns.cbnu.ac.kr/batterymon)
20. Nicholson AJ, Noble BD (2008) Breadcrumbs: Forecasting mobile connectivity. In: Proceedings of 14th ACM international conference on mobile computing and networking (MobiCom'08)
21. Zheng Y, Li Q, Chen Y, Xie X, Ma W-Y (2008) Understanding mobility based on GPS data. In: UbiComp'08: proceedings of the 10th international conference on ubiquitous computing
22. Sohn T, Varshavsky A, LaMarca A, Chen MY, Choudhury T, Smith I, Consolvo S, Hightower J, Griswold WG, de Lara E (2006) Mobility detection using everyday GSM traces. In: UbiComp'06: proceedings of the 8th international conference on ubiquitous computing
23. http://juniperresearch.com/
24. Basney J, Livny M, Buyya R (1999) Deploying a high throughput computing cluster. Prentice Hall, New York
25. Stellner G (1996) Cocheck: Checkpointing and process migration for MPI. In: Proceedings of the 10th international parallel processing symposium, Honolulu, Hawaii, April 15–19
26. Bubendorfer K (1996) Resource based policies for load distribution. Master's thesis. Victoria University of Wellington
27. Khalidi YA, Bernabeu JM, Matena V, Shirriff K, Thadani M (1995) Solaris MC: a multi-computer OS. Sun Microsystems Laboratories Technical Paper, Nov. 1995
28. Ong I, Lim H (2011) Dynamic load balancing and network adaptive virtual storage service for mobile appliances. Int J Inf Proces Syst 7(1):53–62