International Journal of Smart Home
Vol. 7, No. 3, May, 2013

Power and Performance Analysis of Smart Devices

Min Choi

Division of Information and Communication Engineering
Chungbuk National University, 52 Naesudong-ro, Heungdeok-gu, Cheongju,
Chungbuk 361-763, Republic of Korea
mchoi@cbnu.ac.kr

Abstract

The spreading use of Smartphones stimulates growth of mobile computing. However, mobile computing through smartphones poses challenges due to its intrinsic nature of battery capacity, constraints of wireless networks and performance limitation of devices. Most users want their smartphones to be more powerful and high throughput, at the same time, small and light. In this research, we provide the analysis of smartphone in terms of power and performance under two different point of views. Since most smartphones have various components, we develop a queuing model for power analysis so that we can analyze the power dissipation on different components and analyze battery lifetime with different usage behaviors.

Keywords: *Smartphone Power Analysis, Smartphone Performance Analysis*

1. Introduction

Power dissipation occurs even when we are not supposed to, and it leads to power leakage. Due to the inherent nature of smartphone in which operating system is working, power consumption continues even on not actively using. Smartphones play various roles that require many foreground and background jobs, whereas traditional feature phones such as conventional cell phones mainly concern call processing. This property of smartphones results in severe power dissipation [2].

The power leakage on smartphones are as follow [3, 4]: 1) Applications run as background jobs or services: user can install applications to run as background job periodically such as android service. For instance, once we set up google mail sync, android google mail client try to synchronize the content from Google mail and scheduler. 2) Keeping Wi-Fi, bluetooth, GPS enabled when we are not using explicitly: even though the applications that use those sensors are not working, just turning the sensors on consumes power dissipation. For example, the Wi-Fi protocol periodically communicate with access point (AP) the beacon signals. This results in power consumption even on upper layer of Wi-Fi falling into idle stage. 3) Display lightness/contrast: the main concern of power dissipation is due to the LCD brightness. The backlight power minimization can effectively extend battery life for mobile handheld devices [1].

Performance limitation of smartphone stems mainly from its nature of embedded systems. Usually, embedded systems have substantially different design constraints than desktop computing applications. They try to optimize for mobility and lightness rather than for maximum computing performance and throughput. Nevertheless, many consumers want their smartphones to be more powerful and high throughput, at the same time, small and light. Likewise, with this trend and requirement, we need to analyze which factors can affect to mobile system performance, for example, number of processor cores, clock frequency, processor types, and so on.

In terms of smartphone performance, we investigate reviewing some flagship smartphone products on the shelf [5]. Samsung Galaxy S3 [6] was released in May 2012 and is the latest flagship of Samsung smartphones. Even if the technology was way back in Q2 of the 2012, this smartphone has outlasted many other smartphones even the models released in Q4 of the 2012. It has 1.4 GHz quad-core, Exynos chipset with 1 GB RAM. Recently, LG Electronics [7] released their flagship smartphone device in November 2012, the Optimus G. it offers great speed and performance compare with its ancestor models. It has 1.5GHz quad-core, Qualcomm chipset cpu with 2GB RAM. Apple IPhone 5 was released in September and the latest model unit of IPone today. This version runs the latest generation of iOS 6 and has dual-core processor and 1GB of RAM. It has 1.2 GHz dual-core, Apple A6 CPU. The powerful combination of processor and RAM deliver speed, decent graphics unit and running on the latest Android version make these phone stand out when it comes to overall performance.

In this research, we provide the analysis of smartphone power dissipation under two different point of views, usage scenarios and smartphone components. Smartphones have various components in general, such as CPU, GSM, LCD, Backlight, GPS, Wi-Fi, Bluetooth, Graphics, and so on. We develop a queueing model for power analysis so that we can analyze the power dissipation on different components, and analyze battery lifetime with different usage behaviors. Thus, we can discuss the significance of power dissipation by various components and various usage behaviors.

The rest of this paper is organized as follows: Section 2 describes related works. Section 3 and 4 focuses on the details of power and performance analysis on smartphones. Finally, we conclude our work and present future research directions in Section 5.

2. Related Works

Despite the market’s heterogeneity, the nexus of smartphones, wireless broadband, and network-based cloud computing constitutes a perfect storm of opportunity for application developers, luring their attention toward the new platforms.

COMSOL [7] reduces the upfront investment in equipment and technical expertise so dramatically that high-performance computing is now ready for the main stream. Exploratory speedup factors of 6x and 11x in the context of embarrassingly parallel COMSOL Multiphysics computations provide a powerful business justification for Windows HPC. The ability to divide and conquer by distributing the memory required of any problem size allows us to draw conclusions to problems we can’t even fathom today.

Dandelion [9] provides a system implementation on the Maemo Linux smartphone platform and the Rice Orbit body sensor platform. They evaluate Dandelion by implementing real-world applications, and show that Dandelion effectively eliminates the programming gap and significantly reduces the development efforts. We further show that Dandelion incurs a very small overhead.

While successful for complicated mobile applications, such programming styles significantly differ from those used in smartphone application development, making it is difficult for smartphone developers to adopt them. In contrast, Dandelion leverages the simplicity of smartphone-centered body sensor networks and focuses on supporting in-sensor data processing tasks. With this trade-off, Dandelion is able to provide transparency in programming style.

Some systems support programming transparency with an unified OS abstraction or distributed runtime system, mostly based on a virtual-machine approach to hide ISA variances. This approach, however, proves to be inefficient on resource-constrained sensors. In contrast, Dandelion achieves transparency by limiting the senselet functions to data processing and by introducing an extra compilation phase to produce.

3. Analysis of power dissipation

3.1 Power dissipation by various components

Prior to running experiments, we established preliminary experiments consisting of two types of workloads: synthesized workload and real workload. The synthesized workload is to check the power consumption of each device (or sensor). Real workloads are 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 keep always on. 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 2 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 [5] on SAMSUNG Galaxy S [6]. Note that we ran the experiment multiple times, and we always see the same trends in battery-level drops across all runs. The following figures are the preliminary experimental result with synthesized workload.

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

Our experimental environments are as follows: (a) power supply, (b) digital multimeter (True RMS Multimeter), (c) smartphone (SAMSUNG Galaxy S), (d) laptop computer. The rated input voltage/current range of SAMSUNG Galaxy S is 1500mA at 3.7V. Assuming that the voltage difference is stably supplied with 3.7V without drop of electric pressure, measuring only current change with digital multi-meter 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.

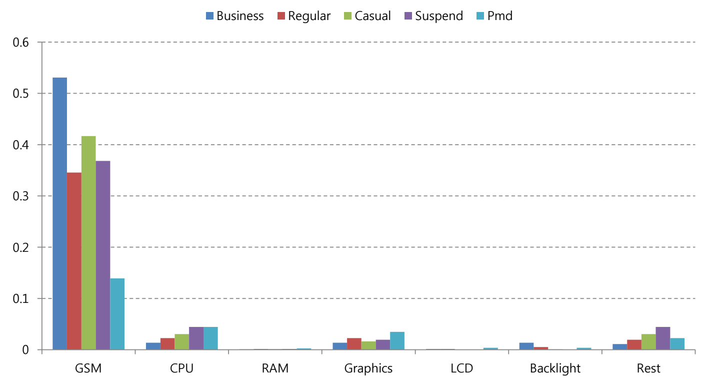

Figure 1. Mean number of requests in the queue on various components

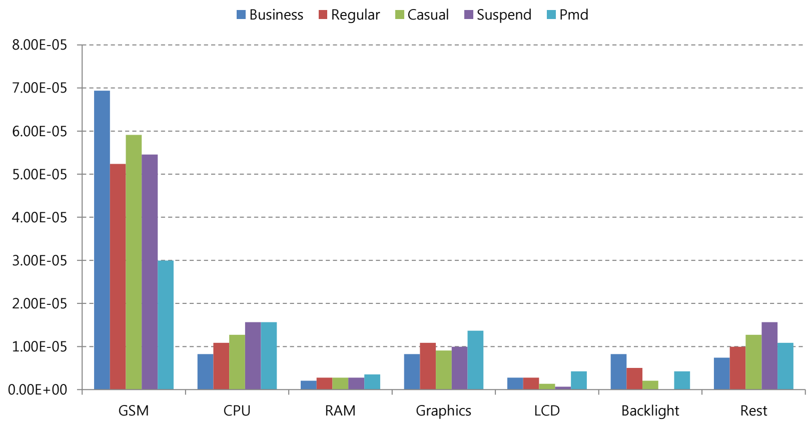

Figure 2. Mean waiting time on various components

Figure 1 shows the mean number of requests in the queue as various components. It shows the fact that relatively larger numbers of requests are queued on GSM. This is because we mainly make use of smartphones for phone call, so relatively more requests are on the GSM components. Figure 2 represents mean number of requests in the queue depending on various component. Since the GSM and CPU are the two of the most busy components in general smartphones, so they are required more mean waiting time rather than other components.

3.2 Power dissipation by usage scenarios

To evaluate the performance, we present a model of queuing network. The model of our REST Open API Web Service architecture is presented in Figure 3. REST Open API Web Service is composed of 3 components comprising: (1) a web server, (2) a REST web server farms, and (3) internet users. As shown in Figure 3, there are a number of components(nodes) that consist of several queues. A request may receive service at one or more queues before exiting from the system. A model in which jobs departing from A arrive at another queue(i.e., the REST Web Server Farm from B1 to B4).

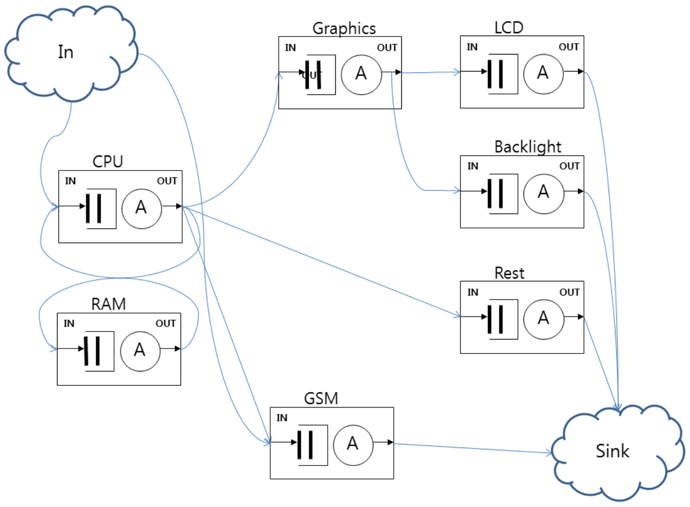

Figure 3. A queuing model for smartphone power dissipation

Requests arrive at the web server A with frequency "In". The initialization process for the request is done at node A. Then, the request proceeds to the component either "CPU" or "GSM" components depending on the type of the request; if the request is for the call processing, it goes to the GSM components. If the request is for just application processing, then it goes to the CPU components. The requests traverse via the RAM, Graphics, LCD, Backlight and Rest. They are finally collected to the Sink node, represented by the components at the right bottom of Figure 3. Our system model is a sort of open queueing network that has external arrivals and departures. The requests enter the system at "IN" and exit at "OUT". The number of requests in the system varies with time. In analyzing an open system, we assume that the throughput is known (to be equal to the arrival rate), and we also assume that there is no probability of incomplete transfer in this system, so there is no retrial path to go back to node A. Now, the CPU components of recent smartphones can have more than one CPU, known as dual-core or quad-core. However, we assume the simple smartphones with single-core in this research.

At each node, let's consider an M/M/1 queue with a processor sharing service discipline. The interarrival times of requests are according to a Poisson process with rate λ and the service times are exponentially distributed and there is only one server. There are no buffer or population size limitations and the service discipline is FCFS. The mean number of requests in the system is given by E[n] = $\sum_{n=1}^{\infty} n p_n = \sum_{n=1}^{\infty} n(1-\rho)\rho^n = \frac{\rho}{1-\rho}$. The mean number of requests in the queue is given by $E[n_q] = \sum_{n=1}^{\infty}(n-1)p_n = \sum_{n=1}^{\infty}(n-1)(1-\rho)\rho^n = \frac{1}{1-\rho}$. When there are no requests in the system, the server is said to be idle; at all other times the server is busy. The probability of n or more requests in the system is (P ≥ requests in system) = $\sum_{n=1}^{\infty} p_{nj} = \sum_{n=1}^{\infty}(1-\rho)\rho^j = \rho^n$. The mean waiting time can be computed using Little's law, which states that mean number in system is arrival rate multiplied by mean response time. That is $E[n] = \lambda E[r]$ or $E[r] = \frac{E[n]}{\lambda} = \left(\frac{\rho}{1-\rho}\right)\frac{1}{\lambda} = \frac{1/\mu}{1-\rho}$. These two expressions are used for performance analysis of M/M/1 model in this research.

Requests arrive from outside following a Poisson process with a certain arrival rate $\lambda > 0$. Each arrival is independently routed to a node B_j within REST web server farm with probability $p_{0B_j} \geq 0$ and $\sum_{j=1}^{J} p_{0Bj} = 1$. Upon service completion at node i, a request may go to another node j with probability p_{ij} or leave the network with probability $p_{i0} = 1 - \sum_{j=1}^{J} p_{ij}$. Hence we have the overall arrival rate to node i, λ_i, including both external arrivals and internal transitions: $\lambda\lambda_i = \alpha p_{0i} + \sum_{j=1}^{J} \lambda_j p_{ji}, i = 1, \dots, J$ then $\lambda = (-P)^{-1}a$.

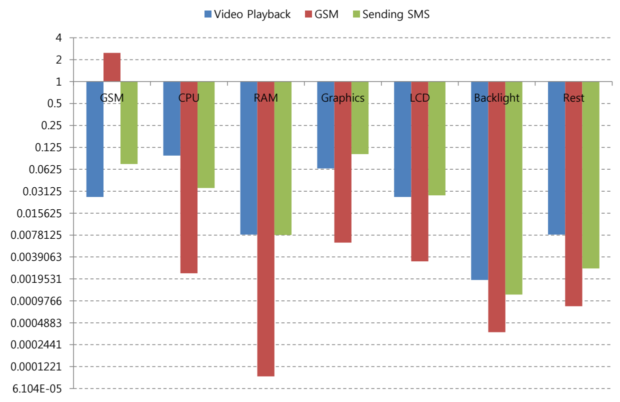

Figure 4. Mean number of requests in the queue as various use cases

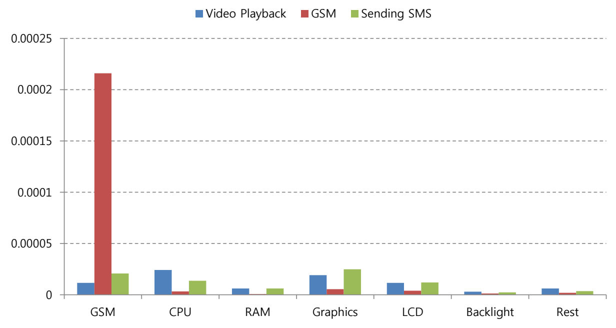

Figure 5. Mean waiting time as various use cases

Figure 4 and Figure 5 presents the mean waiting time and mean number of requests in the queue depending on various usage scenarios. We model each component as the M/M/1 queue because we ignore the case of parallel components, such as multi-cores. These figures show the fact that mean number of request and mean waiting time on which components as different components. The reason why the waiting time and queue length on GSM for GSM usage scenario are quite high is that because we frequently utilize the GSM components during the job. For another case, when it comes to sending SMS to someone, the related components for sending SMS such as Graphics, CPU, GSM, LCD are frequently utilized, so they have higher mean waiting time and mean number of requests during sending SMS use cases.

4. Analysis of system performance on various usage scenarios

In this section, we conducted performance test on smartphones.

All requests submitted must first pass through the web server for providing HTTP service before moving on to the REST web servers, Jersey. Requests arrive at the web server at an average rate of 1000/sec to 15000/sec. To handle the load, the REST web server components may have several parallel clouded or clustered architecture. Table 1 shows the configuration parameters for this performance test. If traffic intensity levels exceed available capacity, customer's calls are not lost.

Table 1. Configuration parameters

System measure	Components A	Component B
Total arrival rate, λ_i	From 1000/sec to 15000/sec	From 1000/sec to 15000/sec
Service rate, μ_i	15000/sec	15000/sec
Multiple number of servers, s_i	1	From 2 to 11
Traffic intensity, ρ_i	λ/μ	$\lambda/s\mu$

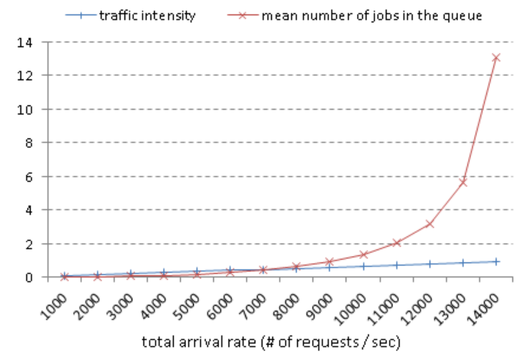

Figure 6. Traffic intensity and mean number of requests in the queue as increasing total arrival rate

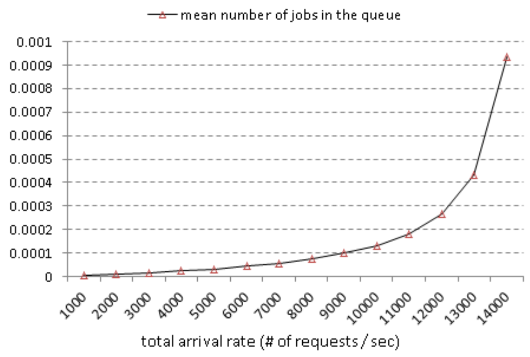

Figure 7. Mean number of requests in the queue as increasing total arrival rate

Figure 6 represents mean number of requests in the queue and traffic intensity at component A. traffic intensity is calculated by the arrival rate over the service rate that means how fast the incoming traffic are serviced on the server. The traffic intensity is a sort of constant on M/M/1 queue (component A is M/M/1 queue). Since the service rate of the Apache web server is 16000 request/sec, the mean number of requests in the queue reaches up to maximum on the total arrival rate is increasing to 15000. Figure 7 shows the mean number of requests in the queue as increasing total arrival rate. And this is the similar to the case of Figure 6.

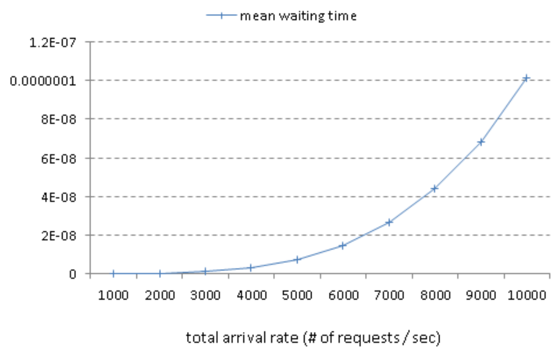

Figure 8. Mean waiting time as increasing total arrival rate

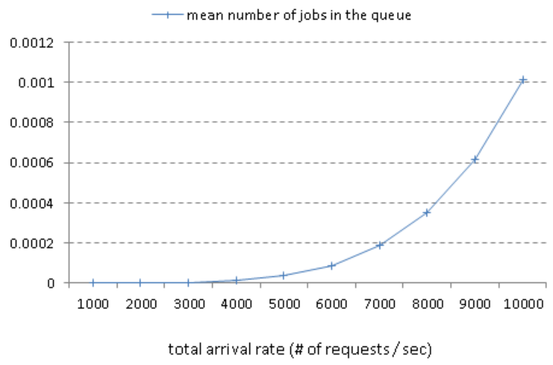

Figure 9. Mean number of requests in the queue as increasing total arrival rate

Figure 8 and Figure 9 presents the mean waiting time and mean number of requests in the queue as increasing total arrival rate at component B. We model the component B as the M/M/m queue. The value m is larger than 1. This means the REST web servers are comprised with multiple Jersey 1.6 servers. These figures show the fact that mean number of request and mean waiting time are increasing as the total arrival rate. The reason why the waiting time and queue length are not reaching the maximum even though the total arrival rate is approaches to the maximum value (15000) is due to the multiple REST web servers. Actually, we carried out this experiment with the four multiple servers of Jersey 1.6 REST web service providers.

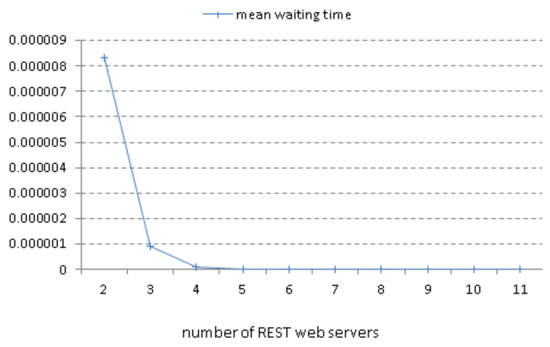

Figure 10. Mean waiting time as increasing number of REST web servers

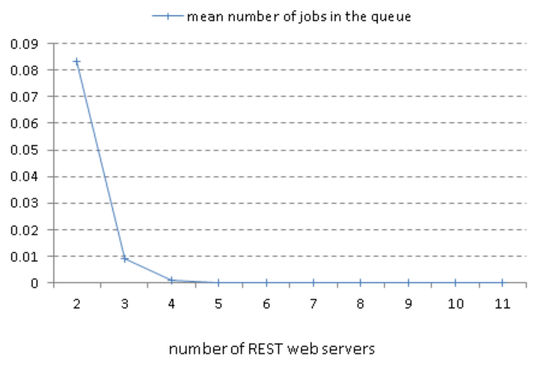

Figure 11 Mean number of requests in the queue as increasing number of REST web servers

Figure 10 and Figure 11 presents the mean waiting time and the mean number of requests by the number of REST web servers. Until now, we just make use of four REST web servers without considering the optimal number of parallelism. So, we carried out the experiment as shown in Figure 10 and Figure 11. From these experiments, we see the fact that 4 or 5 numbers of REST web servers are enough to the current level of workloads.

5. Conclusion

In this research, we provide the analysis of smartphone power dissipation under two different point of views, usage scenarios and smartphone components. In general, smartphones have various components. We develop a queuing model for power analysis so that we can analyze the power dissipation on different components, and analyze battery lifetime with different usage behaviors.

Acknowledgments

This work is supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012-0008105). Corresponding author of this paper is Min Choi (mchoi@cbnu.ac.kr).

References

[1] C. Aaron and H. Gernot, "An Analysis of Power Consumption in a Smartphone", the 2010 USENIX conference on USENIX annual technical conference, **(2010)**.
[2] A. Mahesri and V. Vardhan, "Power consumption breakdown on a modern laptop", In Proceedings of the 2004 Workshop on Power-Aware Computer Systems (Portland, OR, USA, Dec. 2004), B. Falsafi and T. N. Vijaykumar, Eds., vol. 3471 of Lecture Notes in Computer Science, Springer, **(2004)** December, pp. 165-180.
[3] A. Sagahyroon, "Power consumption in handheld computers", In Proceedings of the International Symposium on Circuits and Systems **(2006)** December, pp. 1721-1724.
[4] D. C. Snowdon, E. Le Sueur, S. M. Petters and G. Heiser, "Koala: a platform for OS-level power management", Proceedings of the 4th ACM European conference on Computer systems, **(2009)** April 01-03, Nuremberg, Germany.
[5] V. Hernandez, "2012's Top 8 Smartphones with Best Overall Performance", http://au.ibtimes.com/articles/414498/20121213/2012-s-top-8-smartphones-best-overall.htm.
[6] Galaxy S3, Samsung Electronics, http://www.samsung.com/global/galaxys3/.
[7] Optimus G, LG Electronics, http://www.lgmobile.co.kr/event/optimusG/index.html.

Author

Min Choi received the B.S. degree in Computer Science from Kwangwoon University, Korea, in 2001, and M.S. and Ph.D. degrees in Computer Science from Korea Advanced Institute of Science and Technology (KAIST) in 2003 and 2009, respectively. From 2008 to 2010, he worked for Samsung Electronics as a Senior Engineer. Since 2011 he has been a faculty member of Department of Information and Communication of Chungbuk National University. His current research interests include embedded system, computer architecture, and mobile cloud.