Augmented Reality based Low Power Consuming Smartphone Control Scheme

*Jong-Moon Chung, Taeyoung Ha, Sung-Woong Jo, Taehyun Kyong, and So-Yun Park
School of Electrical and Electronic Engineering, Yonsei University
Seoul, South Korea
[e-mail: jmc@yonsei.ac.kr, taeyoungha@yonsei.ac.kr, cswo02@yonsei.ac.kr, nvidia@yonsei.ac.kr, parksoyun89@yonsei.ac.kr]
*Corresponding author: Jong-Moon Chung

Received May 27, 2017; revised October 17, 2017; accepted October 19, 2017; published October 31, 2017

Abstract

The popularity of augmented reality (AR) applications and games are in high demand. Currently, the best common platform to implement AR services is on a smartphone, as online games, navigators, personal assistants, travel guides are among the most popular applications of smartphones. However, the power consumption of an AR application is extremely high, and therefore, highly adaptable and dynamic low power control schemes must be used. Dynamic voltage and frequency scaling (DVFS) schemes are widely used in smartphones to minimize the energy consumption by controlling the device's operational frequency and voltage. DVFS schemes can sometimes lead to longer response times, which can result in a significant problem for AR applications. In this paper, an AR response time monitor is used to observe the time interval between the AR image input and device's reaction time, in order to enable improved operational frequency and AR application process priority control. Based on the proposed response time monitor and the characteristics of the Linux kernel's completely fair scheduler (CFS) (which is the default scheduler of Android based smartphones), a response time step control (RSC) scheme is proposed which adaptively adjusts the CPU frequency and interactive application's priority. The experimental results show that RSC can reduce the energy consumption up to 10.41% compared to the ondemand governor while reliably satisfying the response time performance limit of interactive applications on a smartphone.

Keywords: Augmented reality, DVFS, ondemand governor

A preliminary version of this paper appeared in ICONI 2016 ("Response time Analysis of Augmented Reality based Smartphone Applications"), and was selected as an outstanding paper. This version exapands on the experimental setup and provides further performance analysis on scheduling a CPU of smartphone. This research was supported by a grant [MOIS-DP-2015-10] through the Disaster and Safety Management Institute funded by Ministry of the Interior and Safety of Korean government.

https://doi.org/10.3837/tiis.2017.10.026 **ISSN : 1976-7277**

1. Introduction

Energy consumption reduction is one of the most important design objectives in battery operated smartphones and mobile devices, especially for smartphones running augmented reality (AR) applications. This is why Android based smartphones use dynamic voltage and frequency scaling (DVFS) techniques, which trade-off processing speed for improved energy efficiency by controlling the frequency and voltage of the central processing unit (CPU). However, a reduced processing speed will lead to slower response time for AR applications, which is a critical issue because AR views tend to change fast as the user's smartphone camera view changes. The response time is usually defined as the time between a user's input and the device's reaction. Most applications running on smartphones are usually interactive applications, which require the user's input, such as games, web browser, and social network services (SNS). For interactive applications, the response time is a very important part of user experience. Among interactive applications, AR is among the most computationally demanding application. It is noted that users commonly feel discomfort when the response time of an interactive application exceeds 150 ms [1]. Therefore, it is important to satisfy the response time limit when a user runs an AR interactive application. In addition, when the AR interactive application runs with other applications, it is more difficult to satisfy the response time limit because the interactive application can be preempted by simultaneously running other applications. Android is the most widely used operating system (OS) for smart devices, which runs on a Linux kernel. Although satisfying the response time limit is an important issue, the Linux kernel does not provide response time information of the smartphone. Therefore, in this paper, a response time monitor is proposed, which periodically measures the response time.

The proposed response time step control (RSC) algorithm is different compared to the other DVFS schemes as it monitors the response time and estimates response time variations based on adaptive control changes made to the CPU frequency and interactive application's scheduling priority to achieve energy consumption minimization while satisfying the target application's response time requirements.

2. Background and Related Work

2.1 Mobile AR

AR is a technology that adds computer generated virtual information or images on the real view of a device's display to provide useful information of selected objects within the image. Whereas a user is involved in a virtual environment when using virtual reality (VR) technology, a user can use the augmented information or interact with virtual images imposed on the real-world environment image through AR technology. Although AR technology was innitially proposed over forty years ago, AR based practical systems have not been widely avaliable due to technical difficuilties in device type and form factor, level of accuracy and reliability, as well as computation load and power consumption [2]. However, recent advancements in hardware and software of mobile computing devices make mobile AR systems (MARSs) and applications possible [3]. In [3], MARS is defined as a system that combines real objects with virtual augmentation, which is based on dynamic three dimentional (3D) objects, and runs in real-time and in mobile mode. In addition, the basic components of

MARS include a computational hardware platform, display, tracking, wireless network, wearable input and interaction, and software [3]. According to the advancements in smartphone specficiations, smartphones have been able to equip with the basic components of MARS. Furthermore, as smartphones are widely used aroud the world, the number of people using smartphone based mobile AR applications is expected to increase significantly [4]. Practical example of a mobile AR application is the navigation system [5]. Navigation is an outdoor-oriented application for mobile AR, and mobile AR based navigation systems use a points of interst, user-created annotations, or graphics based on the global positioning system (GPS) location and magnetometer of a mobile device [6]. Mobile AR based navigation systems are useful to travelers, because these system can provide guidance to a destination as well as information about the surrouding area.

A user experience (UX) is defined as "a person's perceptions and responses that result from the use or anticipated use of a product, system or service" in the ISO standard, where the response time is the most important part of UX for the interactive systems. Furthermore, in recent years, the principal goal in designing an interactive system is achieve satisfaction to users about the UX [6].

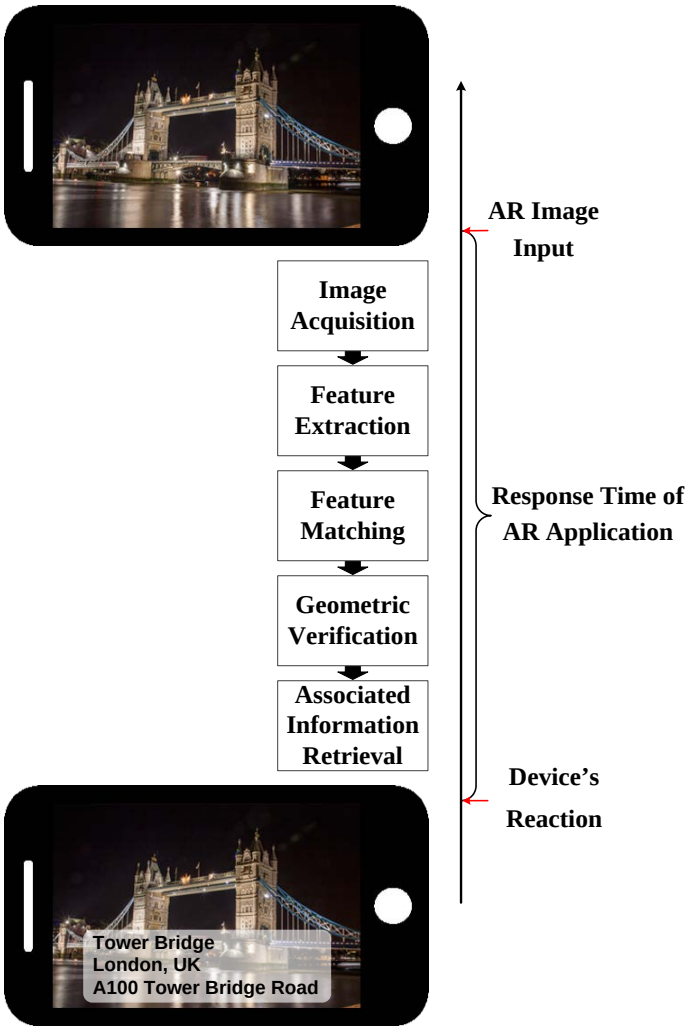

Fig. 1. Effects of CFS and DVFS on Throughput

As introduced in [7], mobile AR procedures can be divided into 5 steps, which are image acquisition, feature extraction, feature matching, geometric verification, and associated information retrieval. As shown in **Fig. 1**, the response time of an AR application can be represented as the time interval between the image input for the image acquisition process and the device's reaction time, which is the time consumed to have the associated information appear on the screen of the smartphone. Therefore, the proposed AR response time monitor is used to observe the time interval between the AR image input and the smartphone's reaction time.

2.2 Completely Fair Scheduler

The CPU scheduler of the Linux kernel has an effect on response time performance because it determines the task execution order and interval [8,9]. The completely fair scheduler (CFS) is the default scheduler of the Linux kernel since Linux kernel version 2.6.23 (which was released in Oct. 2007) to Linux kernel version 4.1.2 (the newest version when this article was written). In addition, all recently released Android based smartphones (from Android 4.2 Jelly Bean to the newest 7.1.2 Nougat) use Linux kernel version 3.4.0 in which CFS is the default scheduler. In CFS, each application has own weight according to its nice value, which is an integer between -20 and 19. When the nice value of an application is decreased by 1, the weight of the application is increased by 1.25 in the Linux kernel version 3.4.0. CFS executes each application according to its time slice, and the time slice of application i is denoted as $S_t(i) = pw_{n^i} / \left(\sum_{\forall j \in S} w_{n^j}\right)$, where w_{n^i} is the weight of application i with nice value n^i , **S** is the set of all applications being processed by the CPU, and p is the period. The period p is defined as $p = \max[6, 0.75N]$ (in units of millisecond) in Linux kernel version 3.4.0, where N is the number of applications in set **S**.

2.3 DVFS

In a smartphone, the display and radio modules consume a large amount of the system's energy [10]. The energy consumption amount depends on the component's power consumption profile as well as the application's display image (and user's brightness settings) and the amount of data traffic the application is required to send and receive [11]. AR applications use all of these resources of the smartphone, and therefore, the energy consumption of AR services is extremely energy consuming. However, the energy consumption of the CPU can be controlled at the kernel and OS level by adjusting the CPU cores' operational frequencies [11,12,13]. Therefore, the proposed scheme focuses on reducing the energy consumption of the CPU while satisfying the response time performance of the application. DVFS in one of the most commonly used CPU energy management techniques. In [14], the power-aware decoder was proposed, which allows OS to adaptively control CPU frequency by delivering information for video decoding based on a low-power microprocessor. In [15], a DVFS scheme for optimizing mobile 3D rendering was proposed. A Linux kernel module called the *governor* uses DVFS to reduce the energy consumption of the CPU. The most commonly used governor is definitely the *ondemand governor*, which is used as the default governor of Android based smartphones. The ondemand governor increases the CPU frequency to its highest value when the CPU load (L_{CPU}) is above its predefined limit, but decreases the CPU frequency step by step when L_{CPU} is below the predefined limit [16]. As DVFS techniques are based only on L_{CPU} [16,17,18], sometimes this results in longer response times for interactive applications. In order to address the problem in load based DVFS schemes, several approaches have been proposed. Cinder [19] was proposed, which manages

the energy usage at the OS level for mobile phones and other energy-constrained computing devices by restricting the power level. In cases when then Cinder regulates the power consumption level significantly, the smartphone's response time may become very long.

3. Effects of CFS and DVFS on Throughput

Fig. 2 shows how CFS and DVFS affect the throughput of each application. Throughput is defined here as the amount of executed task of the application in a given period by the CPU. The throughput is influenced by both the processing speed and executed time. The processing speed of a CPU is proportional to the CPU frequency and it is represented as the height of the blocks in **Fig. 2**, where as the CPU frequency is increased, the height of the block is taller. The executed time is proportional to the weight of each application, which is represented as the width of the block in **Fig. 2**. The executed time is represented based on the time slice in CFS and it is determined by the weight of each application.

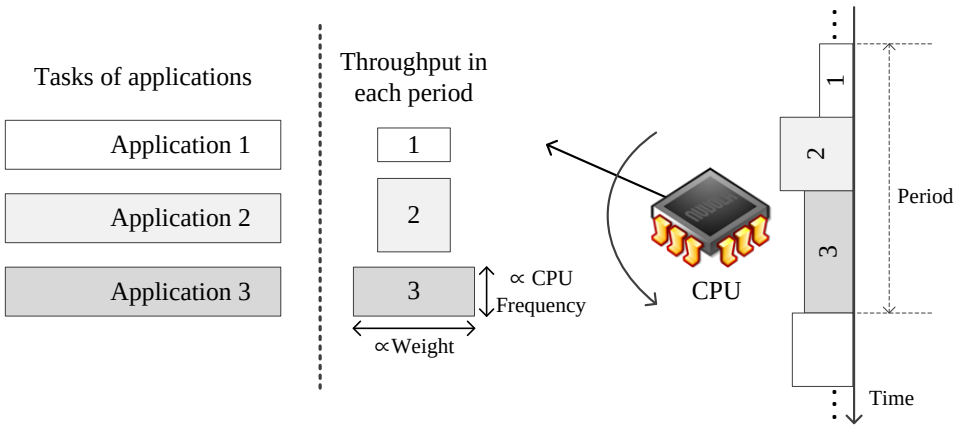

Fig. 2. Effects of CFS and DVFS on Throughput

CFS executes each application in a round-robin fashion. For this reason, an application experiences preemption when other applications are executed in the same period.

4. Response Time Monitor

Fig. 3 represents the response time monitor architecture that is used in the proposed RSC scheme. The response time is measured as the time interval between the Start time and Stop time. The Start time is the time instant when the input event occurs (i.e., user's input) and the Stop time is the time instant when the results of the input event updates the display (i.e., device's reaction). The Timer is used for recording the time information of events. When a user runs an interactive application, the input event occurs by touching the screen of the smartphone. When an input event occurs, the onTouch() function of View.OnTouchListener is used to obtain data, which acts as the input catcher. WindowManagerService is executed regularly in the Framework in order to manage the window. At this time, the Framwork's InputManager is executed by InputManage.Start() in WindowManagerService. By the InputManager's nativeStart function, the Android_server_InputManager_nativeStart of NativeInputManager based on the JAVA Native Interface (JNI) is called. The Android_server_InputManager_nativeStart's start() executes the native code's InputManager::start(), and here, the InputReader is executed by the run function. When an

input is entered, the device driver stores the value in the EventHandler's input_event structure. The Android OS receives this input key value through the InputReader's EventHub::getEvent(), and after comparing it with its key layout file, it updates the key value of the Framework. By using this function, the input catcher perceives the input event and the Timer records the Start time using the Event.getDownTime() function, which returns the current time. Then, the interactive task is executed according to the input information and the results are returned to the application. As the results need to be updated on the display, the application sends its result to the SurfaceFlingerClient using the Framework's surface. The SurfaceFlingerServer collects data that is sent from the SurfaceFlingerClient and the data is sent to the frame buffer (FB) driver through opengl and the hardware abstraction layer (HAL). Finally, the FB driver displays the data on the display. When the layout state of the display is changed, the OnGlobalLayout() function is invoked. By using this function, the output catcher perceives the changes in the display and the Timer records the Stop time with the System.nanoTime() function which returns the current time in units of nanoseconds.

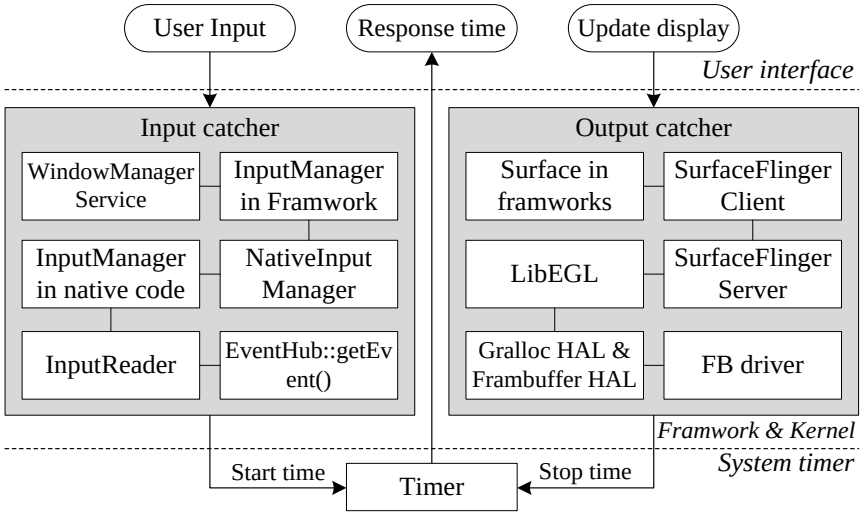

Fig. 3. Response time monitor implementation architecture

5. RSC Algorithm

5.1 Response Time Difference Estimation

Service time is the time that is required by the processing of the CPU to run the target application. The service time $T_S(f)$ of an interactive application at the CPU frequency f can be represented as $T_S(f)=\{(\alpha/f)+\beta\}$, where α/f is the CPU frequency-dependent workload and β is the CPU frequency-independent workload [20,21]. By using this equation, the service time of the AR interactive application is estimated for various CPU frequencies. The weight of application i with nice value n^i is $w_{n^i}=1.25^{-n^i}w_0$, which represents the weight of the running applications with nice value 0. The nice value of the AR interactive application is denoted as n^* and its corresponding weight is $w_{n^*}=1.25^{-n^*}w_0$. The response time of the AR interactive application $T_R(f_{CPU},n^*)$ is a function of both the CPU operational frequency f_{CPU} and n^*. $T_R(f_{CPU},n^*)$ is influenced by the AR interactive application's service

time $T_S(f)$ and preemption rate. The preemption rate is the rate of the target AR application being preempted by other applications during its service time. The preemption rate of the AR interactive application can be derived from the ratio of the weight distribution of all processing tasks that share the CFS in reference to the AR interactive application's weight (i.e., $\left(w_{n^*} + \sum_{\forall i \in S'} w_{n^i}\right) / w_{n^*}$), where **S**' is the set of all tasks being processed by the CPU, excluding the target AR application. In addition, the preemption rate of the AR interactive application is affected by the CPU load of the background applications (L_B). This is because preemption of the AR interactive application occurs when the background applications are simultaneously running with the AR application, and L_B represents how often the background applications occupy the CPU resources. Therefore, the AR interactive application will experience a Δ_R step difference in its response time if the AR interactive application's nice value is changed to n^*_y (from its current nice value n^*) or the CPU frequency is changed to f_x (from the smartphone's current CPU frequency f_{CPU}). The step difference in response time Δ_R is represented in (1).

$$\Delta_R = T_S(f_x)\frac{w_{n^*_y} + L_B \sum_{\forall i \in S'} w_{n^i}}{w_{n^*_y}} - T_S(f_{CPU})\frac{w_{n^*} + L_B \sum_{\forall i \in S'} w_{n^i}}{w_{n^*}} \quad (1)$$

If the CPU were to operate at f_x Hz, then the AR interactive application's (with nice value n^*_y) response time (i.e., $T_R(f_x, n^*_y)$) can be obtained from the current response time $T_R(f_{CPU}, n^*)$ added to the step difference in response time Δ_R as shown in (2).

$$T_R(f_x, n^*_y) = T_R(f_{CPU}, n^*) + \Delta_R \quad (2)$$

5.2 RSC Algorithm

In a smartphone, the number of applicable CPU frequencies and application nice values are limited. The set of selectable CPU frequencies are defined in **F**={$f_1, f_2, \ldots, f_x, \ldots, f_{max-1}, f_{max}$} and the set of selectable nice values are defined in **N**={$n_1, n_2, \ldots, n^*_y, \ldots, n_{max-1}, n_{max}$}. The proposed RSC algorithm's pseudo code is presented in **Fig. 4**.

The algorithm first sets the CPU to its maximum frequency and the nice value of the AR interactive application to 0, which is the default value of CFS (step 1). Since the power consumption of the smartphone is reduced for lower CPU frequencies, the algorithm searches for the lowest CPU frequency starting from the highest frequency (f_{max}), where the search will continue in a sequence of procedures comparing the estimated response time of (2) (using the response time step difference (step 4) of (1) at a lower CPU frequency (step 3)) to the predefined response time limit L_R (step 5). If a suitable lower frequency is found, then that frequency is used as the new CPU frequency (step 5.1) and the search for a lower frequency will be attempted again (step 5.2). When a lower CPU frequency cannot be found to satisfy the $T_R(f_x, n^*_y) \le L_R$ condition (step 6), then based on the frequency next lower to f_{CPU} (i.e., $f_x = f_{CPU-1}$) the response time step difference (step 6.3) will be computed and will be tested if it can satisfy $T_R(f_x, n^*_y) \le L_R$ (step 6.4) using a smaller nice value (i.e., higher priority) (step 6.2). If a suitable nice value n^*_y at f_x is found to satisfy $T_R(f_x, n^*_y) \le L_R$, then the frequency f_x is

used as the new CPU frequency and the nice value n_y^* is applied to the interactive application (step 6.4.1) and the search for a lower frequency will be attempted again (step 6.4.2). Through these procedures, the RSC algorithm will find the lowest frequency among **F** (and the required nice value among **N**) that can satisfy the response time limit L_R.

The proposed response time step control (RSC) algorithm is different compared to the other DVFS schemes as it monitors the response time and estimates response time variations based on adaptive control changes made to the CPU frequency and interactive application's scheduling priority to achieve energy consumption reduction while satisfying the AR application's response time requirements.

BEGIN

1. SET $f_{CPU} \leftarrow f_{max}$, $f_x \leftarrow f_{max}$, $n^*=0$, $n_y^* \leftarrow n^*$
2. IF $f_{CPU} = f_1$
   2.1. GOTO END
3. SET $f_x \leftarrow f_{x\text{-}1}$
4. COMPUTE Δ_R, $T_R(f_x, n_y^*)$
5. IF $T_R(f_x, n_y^*) \le L_R$
   5.1. SET $f_{CPU} \leftarrow f_x$
   5.2. GOTO step 2
6. ELSE
   6.1. IF $n_y^* = n_1$
      6.1.1. GOTO END
   6.2.SET $n_y^* \leftarrow n_{y-1}^*$
   6.3. COMPUTE Δ_R, $T_R(f_x, n_y^*)$
   6.3.4. IF $T_R(f_x, n_y^*) \le L_R$
      6.4.1. SET $f_{CPU} \leftarrow f_x$, $n^* \leftarrow n_y^*$
      6.4.2. GOTO step 2
   6.5. ELSE GOTO step 6.1

END

Fig. 4. Pseudo code of the RSC algorithm

6. Experimental Environment and Results

The experiments were conducted on a Samsung Galaxy S4 LTE-A smartphone based on the Linux kernel 3.4.0, which uses a Qualcomm Snapdragon 800 MSM 8974 quad-core processor 2.3 GHz CPU and Android 4.2.2 Jelly Bean operating system. The power consumption was measured using a *Monsoon FTA22D power monitor* and *Agilent 66321D mobile communications DC source with battery emulator* for the CPU loads of the background applications L_B=0%, 10%, 20%, and 30%. The AR interactive application was selected to be *Asphalt 8* and *Facebook*, which is in the high ranks of racing games and SNS in the Google Play Store, respectively. Input events for each interactive application occurred as driving a car in Asphalt 8 and scrolling pages in Facebook. For the experiments, the brightness of the smartphone was set to its maximum, Wi-Fi and Bluetooth were turned off, and LTE-A was

turned on. The experiments were conducted when the network conditions were very good (i.e., minimal round trip time (RTT) conditions), and therefore, the results of the experiments are majorly compute-bound influenced (rather than I/O-bound influenced). In addition, the response time limit was set to L_{R_1} =125 ms and L_{R_2} =150 ms, and the sets of selectable CPU frequencies and nice values are **F**={300, 422.4, 652.8, 729.6, 883.2, 906, 1036.8, 1190.4, 1267.2, 1467.6, 1574.4, 1728, 1958.4, 2265.6} (all in units of MHz) and **N**={-20, …, 0,…, 19}, respectively.

The system parameters of the smartphone and applications for implementing the RSC algorithm are obtainable in the following way. Linux provides CPU frequency information at */sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq* and information of the running applications at */proc/{pid}/stat*, which include the process identification (PID) of the applications and the PID stat data. PID stat data includes information of the nice value, execution time of process on user mode (*utime*), and kernel mode (*stime*). Service time of the interactive application from time t_1 to time t_2 can be computed from $T_S(f) = \{(utime_{t_2} + stime_{t_2}) - (utime_{t_1} + stime_{t_1})\}$ /HZ in unit of ms, where HZ is a constant value which is 100 in Linux kernel version 3.4.0. By using these obtained values, RSC was implemented on the given experimental environment.

In order to evaluate the accuracy of (1), the response time step differences were measured for various L_B cases. In **Fig. 5** and **Fig. 6**, the estimated response time step difference in (1) was compared with the experimental results. **Fig. 5** and **Fig. 6** show the accuracy of (1) for different nice values with maximum CPU frequency and for different CPU frequencies based on the default nice value, respectively. For all tested cases, a good match between the estimated response time step difference model and the actual measured values were confirmed. The average error of **Fig. 5** and **Fig. 6** are 0.103 ms and 0.698 ms respectively, and in other cases (for different frequencies and nice values) the errors between measured values and estimated values did not exceed 0.837 ms.

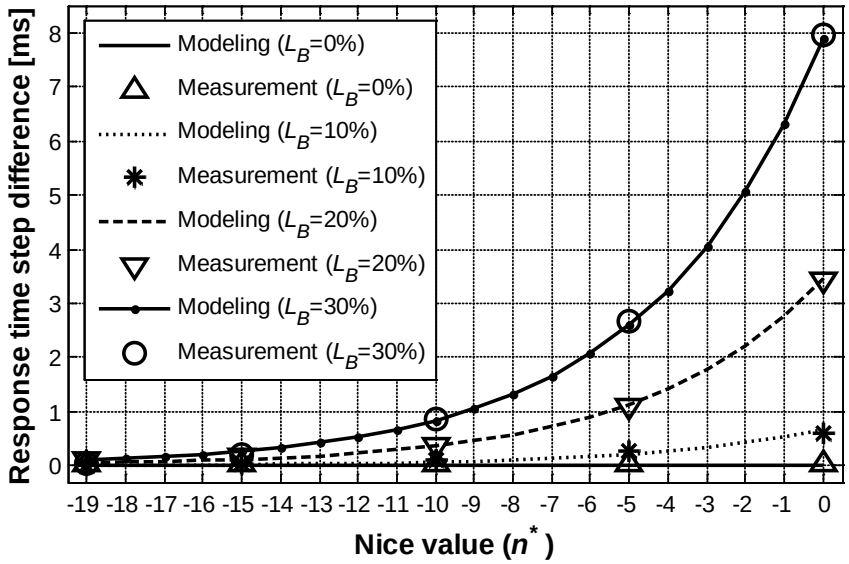

Fig. 5. Accuracy of the response time step difference of (1) with nice value variations.

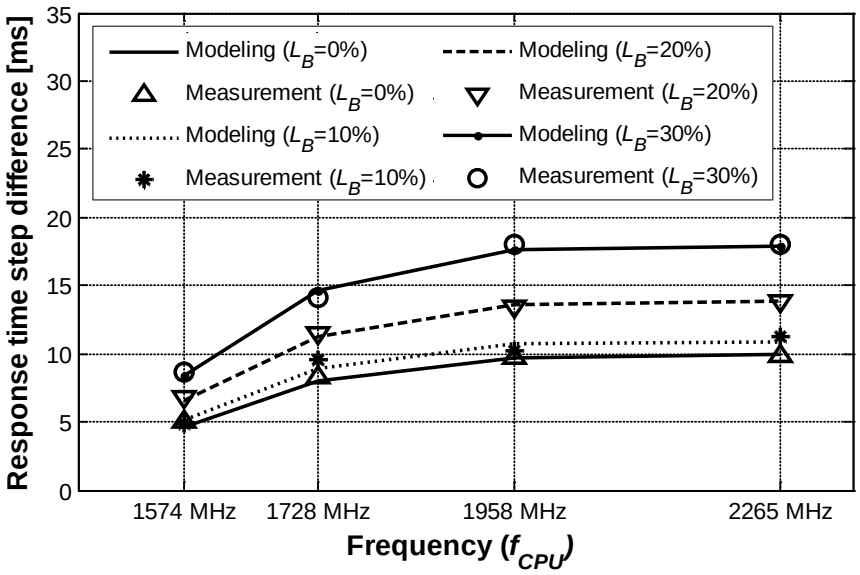

Fig. 6. Accuracy of the response time step difference of (1) with CPU frequency variations.

In order to compare the energy consumption, the energy model of [12] was considered in the measurement analysis. Based on repeated experiments conducted on the smartphones, the applications were invoked in a periodic fashion, where the energy consumption average over a single period is used in the comparison. The system energy E_{total} is obtained from the combination of dynamic energy and static energy in the form of $E_{total} = E_D + (T_{\max} - T_R)P_{idle}$ [12]. The dynamic energy E_D is a function of the CPU clock frequency and includes the system components, such as, CPU, main memory, and I/O devices [12,13]. The static energy consumption profile can be obtained from $(T_{\max} - T_R)P_{idle}$, where $T_{\max}$ is the maximum response time, T_R is the current response time of the system, and P_{idle} is the idle time power consumption [12], which was measured using the Monsoon FTA22D and Agilent 66321D. P_{idle} is required for purposes such as keeping the system clock running and maintaining the basic circuits [13]. After experiments on RSC were conducted, the same average power level of RSC was used as the power level in Cinder, such that a fair response time performance comparison could be made. **Fig. 7** presents the energy savings percentage (%) of the Cinder scheme [19] and the proposed RSC scheme (respectively) in reference to the energy consumption of the ondemand governor, measured on the same Samsung Galaxy S4 smartphone with all equal test conditions. Since the amount of energy consumption of the smartphone's display and other modules were all the same for all test scenarios, the energy savings percentage of Cinder and RSC presented in **Fig. 7** can be considered as the savings in CPU energy. **Fig. 8** presents the response time performance of RSC and Cinder based on response time limits and applications. In **Fig. 7**, for the case where L_B is 0%, the energy is overused due to an excessively enlarged response time. However, energy savings are obtained when L_B is 10%, 20%, and 30%. The results show that RSC can provide a savings in energy consumption up to 10.41% and 10.22% for Asphalt 8 and Facebook, respectively, compared to the ondemand governor. However, for the 40% and higher background load cases, the CPU

operates at its maximum frequency, and therefore, no energy savings were obtainable from RSC or Cinder. In addition, when L_R is small (e.g., $L_R = L_{R_1}$), the energy saving is less than when L_R is large (e.g., $L_R = L_{R_2}$). This is because, a reduced L_R requires the smartphone to operate at a higher frequency. In **Fig. 8**, when L_B is 20% and 30%, the response time of the smartphones using the ondemand governor and Cinder exceeds L_{R_1} and L_{R_2}. However, when L_B =10%, 20%, and 30%, RSC can maintain a response time below the response time limit.

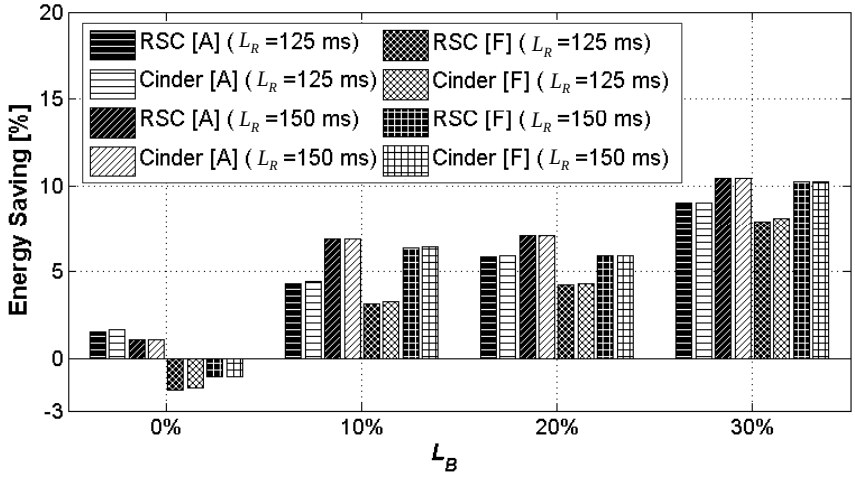

Fig. 7. Energy savings performance of the RSC algorithm

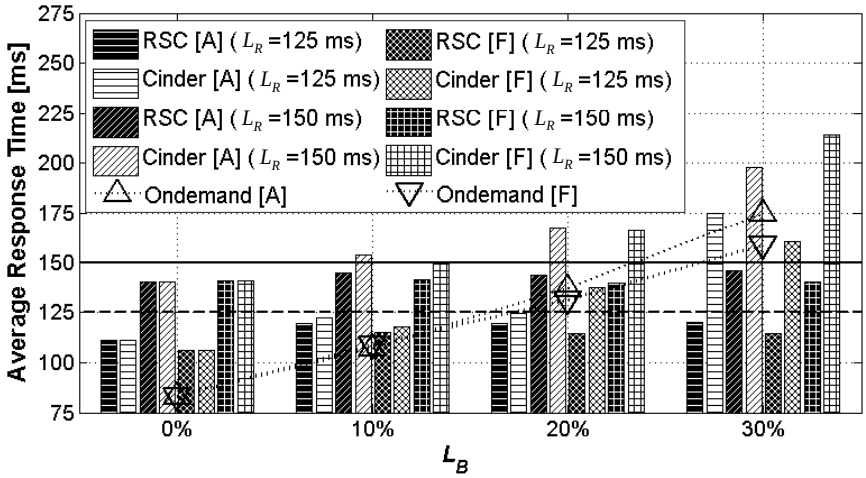

Fig. 8. Response time performance of the RSC algorithm

8. Conclusions

In this paper, the RSC algorithm is proposed, which controls both the CPU frequency and priority of the AR interactive application by estimating the difference in response time. RSC gradually reduces the CPU frequency to the minimum CPU frequency which does not exceed L_R. Experimental results show that RSC can save up to 10.41% of energy when compared to

the ondemand governor for the 30% background load case. Furthermore, by adapting the priority of the AR interactive application, the RSC algorithm is able to keep the response time of the smartphone under the desired response time limit L_R even for the cases where the background applications have a 30% load condition.

References

[1] N. Tolia, D. G. Andersen, and M. Satyanarayanan, "Quantifying Interactive User Experience on Thin Clients," *IEEE Computer*, vol. 39, no. 3, pp. 46-52, March, 2006. Article (CrossRef Link).

[2] F. Zhou, H. B.-L. Duh, and M. Billinghurst, "Trends in Augmented Reality Tracking, Interaction and Display: A Review of Ten Years of ISMAR," in *Proc. of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 193-202, September 15-18, 2008. Article (CrossRef Link).

[3] G. Papagiannakis, G. Singh, and N. Magnenat-Thalmann, "A survey of mobile and wireless technologies for augmented reality systems," *Computer Animation and Virtual Worlds*, vol. 19, no. 1, pp. 3-22, 2008. Article (CrossRef Link).

[4] J. Wither, S. DiVerdi, and T. Höllerer, "Annotation in outdoor augmented reality," *Computers & Graphics*, vol. 33, no. 6, pp. 679-689, December, 2009. Article (CrossRef Link).

[5] T. Höllerer and S. Feiner, "Mobile augmented reality," *Telegeoinformatics: Location-Based Computing and Services*, Taylor and Francis Books Ltd., London, 2004.

[6] T. Olsson and M. Salo, "Online User Survey on Current Mobile Augmented Reality Applications," in *Proc. of the 10th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 75-84, October 26-29, 2011. Article (CrossRef Link).

[7] J.-M. Chung, Y.-S. Park, J.-H. Park, and H. Cho, "Adaptive Cloud Offloading of Augmented Reality Applications on Smart Devices for Minimum Energy Consumption," *KSII Transactions on Internet and Information Systems*, vol. 9, no. 8, pp. 3090-3102, August, 2015.

[8] S. Wang, Y. Chen, W. Jiang, P. Li, T. Dai, and Y. Cui, "Fairness and Interactivity of Three CPU Schedulers in Linux," in *Proc. of Int. Conf. on Embedded and Real-Time Computing Systems and Applications*, pp. 172-177, August 24-26, 2009. Article (CrossRef Link).

[9] S. Huh, J. Yoo, and S. Hong, "Improving Interactivity via VT-CFS and Framework-Assisted Task Characterization for Linux/Android Smartphones," in *Proc. of Int. Conf. on Embedded and Real-Time Computing Systems and Applications*, pp. 250-259, August 19-22, 2012. Article (CrossRef Link).

[10] A. Carroll and G. Heiser, "An Analysis of Power Consumption in a Smartphone," in *Proc. of USENIX Annual Technical Conference*, pp. 1-14, June 23-25, 2010.

[11] C. Yoon, D. Kim, W. Jung, and C. Kang, "Appscope: Application Energy Metering Framework for Android Smartphone Using Kernel Activity Monitoring," in *Proc. of USENIX Annual Technical Conference*, pp. 36-49, June 13-15, 2012.

[12] D. C. Snowdon, E. Le Sueur, S. M. Petters, and G. Heiser, "Koala: A Platform for OS-Level Power Management," in *Proc. of the 4th ACM European Conf. on Computer systems*, pp. 289-302, April 1-3, 2009. Article (CrossRef Link).

[13] V. Devadas and H. Aydin, "On the Interplay of Voltage/Frequency Scaling and Device Power Management for Frame-based Real-time Embedded Applications," *IEEE Transactions on Computers*, vol. 61, no. 1, pp. 31-44, January, 2012. Article (CrossRef Link).

[14] J. Pouwelse, K. Langendoen, and, H. Sips, "Dynamic Voltage Scaling on a Low-Power Microprocessor," in *Proc. of the 7th Annual Int. Conf. on Mobile computing and networking*, pp. 251-259, Jul. 2001.

[15] B. C. Mochocki, K. Lahiri, and S. Cadambi, "Signature-Based Workload Estimation for Mobile 3D Graphics," in *Proc. of the 43rd Annual Design Automation Conf.*, pp. 592-597, Jul. 2006.

[16] V. Pallipadi and A. Starikovskiy, "The Ondemand Governor," in *Proc. of Ottawa Linux Symposium*, pp. 139-152, July 19-20, 2006.

[17] D. Brodowski, "CPUFreq Governors," https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt, 2013.
[18] "Interactive Governor," https://lkml.org/lkml/2012/2/7/483, 2012.
[19] A. Roy, S. M. Rumble, R. Stutsman, P. Levis, D. Mazieres, and N. Zeldovich, "Energy Management in Mobile Devices with the Cinder Operating System," in *Proc. of the 6th Conf. on Computer Systems*, pp. 139-152, April 10-13, 2011. Article (CrossRef Link).
[20] M. Marinoni and G. Buttazzo, "Elastic DVS Management in Processors with Discrete Voltage/Frequency Modes," *IEEE Transactions on Industrial Informatics*, vol. 3, no. 1, pp. 51-62, February, 2007. Article (CrossRef Link)
[21] S.-W. Jo, T. Ha, T. Kyong, and J.-M. Chung, "Response Time Constrained CPU Frequency and Priority Control Scheme for Improved Power Efficiency in Smartphones," *IEICE Transactions on Information and Systems*, vol. E100-D, no.1, pp. 65-78, January, 2017.

Dr. Jong-Moon Chung received B.S. and M.S. degrees in electronic engineering from Yonsei University, Seoul, Korea, in 1992 and 1994, respectively, and Ph.D. degree in electrical engineering from the Pennsylvania State University, University Park, PA, USA, in 1999. Since 2005, he has been a Professor in the School of Electrical & Electronic Engineering, Yonsei University, Seoul, Republic of Korea (ROK). From 1997 to 1999, he served as an Assistant Professor and Instructor in the Department of Electrical Engineering, Pennsylvania State University, University Park. From 2000 to 2005, he was with the School of Electrical & Computer Engineering, Oklahoma State University (OSU), Stillwater, OK, USA as a Tenured Associate Professor and Director of the OCLNB and ACSEL labs. His research is in the area of smartphone design, network scheduler design, M2M, IoT, AR, CR, SDN, NFV, MANET, VANET, WSN, satellite & mobile communications, and broadband QoS networking. In 2000 he received the First Place Outstanding Paper Award at the IEEE EIT 2000 conference. In 2003 and 2004, respectively, he received the Distinguished Faculty Award and the Technology Innovator Award, both from OSU. As an Associate Professor at OSU, in October 2005, he received the Regents Distinguished Research Award and in September the same year he received the Halliburton Outstanding Young Faculty Award. In 2008 he received the Outstanding Accomplishment Professor Award from Yonsei University. In 2012 he received the ROK Defense Acquisition Program Administration (DAPA) Award. He is a senior member of the IEEE, member of the IET and IEICE, and a life member of the HKN, KSII, IEIE, and KICS. He has served as the General Co-Chair of IEEE MWSCAS 2011, Local Chair and TPC Co-Chair of IEEE VNC 2012, and Local Chair of IEEE WF-IoT 2014. He is Co-EiC of the KSII TIIS, Editor of the IEEE Transactions on Vehicular Technology, Section Editor of the ETRI Journal, and Editor of the ICT Express.

Taeyoung Ha received his B.S. degree from the School of Electrical and Electronic Engineering, Yonsei University, Korea, in 2012. He is currently a Ph.D. candidate in the School of Electrical and Electronic Engineering and a research member of the Communications & Networking Laboratory (CNL). His research focuses on low power embedded systems, real-time systems, mobile and ad hoc wireless networks, and MAC protocols.

Sung-Woong Jo received his B.S. degree from the School of Electrical and Electronic Engineering, Yonsei University, Korea, in 2011. He is currently a Ph.D. candidate in the School of Electrical and Electronic Engineering and a research member of the CNL. His research focuses on low power embedded systems, real-time systems, QoS scheduling, and wireless networks.

Taehyun Kyong received his B.S. and M.S. degree from the School of Electrical and Electronic Engineering, Yonsei University, Korea. He is currently working with LG Electronics. His research focuses on low power designs, computer architecture, and wireless sensor networks.

So-Yun Park received his B.S. degree from the School of Electrical and Electronic Engineering, Yonsei University, Korea, in 2015. She is a MS candidate in the School of Electrical and Electronic Engineering and a research member of the CNL.