Real-Time Scheduling Algorithms and Battery Consumption of Mobile Devices

Raquel Elizondo*, Martín Flores*, Jose Salazar*, Oscar Rodríguez* and Nelson Méndez*
School of Computer Science
Instituto Tecnológico de Costa Rica, Cartago, Costa Rica
*Email: {rackelelizondo, mfloresg, bimbosalazar, oscar.rodar, n.mendezmontero}@gmail.com

***Abstract*—One of the most increasing areas in application development is real time applications, as time goes by and technology develops more powerful devices the applications are now requested by users as real time applications, extended reality applications, and more complex applications such as online banking and more that requires more complex implementations every time. As much as we as users like these new applications and the new possibilities we have with them, there is always a concern regarding this kind of applications in mobile devices: the energy consumption. For this applications to run and perform as expected, a considerable amount of energy is needed, for these applications the constant communication with peers and/or main services is essential, and for that live interaction the device needs to spend more energy than a plain classic application, specifically for jobs that the processor executes periodically to keep the live interaction as expected. There are some approaches for this problem that involve designs of algorithms for scheduling these kind of jobs with the objective of saving energy, or at least spend it wisely. In this paper we discuss some of the algorithms that have been proposed to mitigate this issue and keep the user experience the best possible by using battery energy in a smart way but still guaranteeing a very good performance of real time applications.**

I. Introduction

NOWADAYS real-time applications are taking over mobile devices more than ever and, although this brings a lot of new opportunities, it brings challenges too. There is a plenty of examples of applications that includes collective, social, distributed and even virtual/augmented reality features out there and, it just a matter to take a look to a smartphone of an average user to see several of them in action. But, besides the success behind laptops and cell phones as the mobile devices, currently there are many other hardware architectures which has mobile capabilities as well and, in which the implementation of new and interactive applications have increased in recent years. Weareable technolgy and ARM architectures are good examples of this.

Thanks to hardware advances, both companies and users are pushing to provide and experience better applications on mobile devices but this still has big impact in its performance, particularly in battery conpsumption. Research groups have focused on optimizing the hardware of mobile devices, as well as their middleware increasing both the devices' uptime and the user satisfaction but the greater demand in performance for both mobile device hardware and its applications conflicts with the desire for longer battery life.

Users are craving for longer battery life, some even claim that the next big thing in technology ought be better batteries [7]. Taking as example mobile applications, a study conducted by [17] analyzed 9 millions user comments and 27,000 mobile applications under the Google Play store and demostrated that users of mobile applications are interested in enery-efficient applications and energy-inefficient applications lead to frustrated users to negative feedback and lower application ratings. Some other relevants results of the study revealed that: more than 18% of the analyzed applications have user feedback comprising comments on energy consumption problems, energy-efficiency issues negatively influence the grades users give for mobile apps in market places and that free apps do not have more energy consumption problems than payed apps.

The energy use of a typical laptop computer, smartphone and a tablet is dominated by the backlight and display, disks and CPU. They use a number of techniques to reduce the energy consumed by the disk and display, primarily by turning them off after a period of no use. Power consumed by the CPU is becoming more significant because of the type of the applications they run now: data-intensive, multimedia, games and collaborative applications that need to be in a constant sync with networks, sensors, and other devices. Dynamic voltage scaling (DVS) is a common mechanism to save CPU energy. It exploits an important characteristic or CMOS[1]-based processors: the maximun frequency scales almost linearly to the voltage, and the energy consumed per cycle is proportional to the square of the voltage. A lower frequency hence enables a lower voltage and yields a quadratic energy reduction [3]. The major goal of DVS is to reduce energy by as much as possible without degrading appplication performance. The efectiveness of DVS techniques is, therefore, dependent on the ability to predict application CPU demands – overestimating them can waste CPU and energy resources, while understimating them can degrade aplication performance [19].

A real-time operating system has a well-specified maximum time for each action that it performs to support applications with precise timing needs. Systems that can guarantee these maximum times are called hard real-time systems. Those that meet these times most of the time are called soft real-time

This document was proposed as part of the *Advanced Topics in Operative Systems* course at Instituto Tecnológico de Costa Rica. First Semester, 2017.
Abstract and References revised on March 20th, 2017.
First draft revised on May 8th, 2017.

[1]CMOS: Complementary metal–oxide–semiconductor

systems. Deploying an airbag in response to a sensor being actuated is a case where you would want a hard real-time system. Decoding video frames is an example of where a soft real-time system will suffice

In this paper, we present [a set of] [the following] real-time scheduling algorithms [A, B, C] that have been proposed to improve energy-saving for mobile devices. We discuss their main features

II. Real-Time Task Scheduling Algorithms

Several schemes of classification of real-time task scheduling algorithms exist. A scheme in [11] classifies the real-time task scheduling algorithms based on how the scheduling points (the points on time line at which the scheduler makes decisions regarding which task is to be run next) are defined. According to this classification scheme, the three main types of schedulers are: clock-driven, event-driven, and hybrid. Important examples of event-driven schedulers are Earliest Deadline First (EDF) and Rate Monotonic analysis (RM). Event-driven schedulers are more sophisticated and usually more proficient and flexible than clock-driven schedulers. These are more proficient because they can feasibly schedule some task sets which clock-driven schedulers can not. These are more flexible because they can feasibly schedule sporadic and aperiodic tasks in addition to periodic tasks, whereas clock-driven schedulers can satisfactorily handle only periodic tasks [11]. Event-driven scheduling of real-time tasks was a subject of intense research during early 1970s, leading to the publication of a large number of research results, of which the following two popular algorithms are the essence of all EDF and RM [9].

A. Earliest Dealine First (EDF)

Earliest deadline scheduling is simple in concept. Every process tells the operating system scheduler its absolute time deadline. The scheduling algorithm simply allows the process that is in the greatest danger of missing its deadline to run first. Generally, this means that one process will run to completion if it has an earlier deadline than another. The only time a process would be preempted would be when a new process with an even shorter deadline becomes ready to run. To determine whether all the scheduled processes are capable of having their deadlines met, the following condition must hold:

$$\sum_{i=1}^{n} \frac{C_i}{P_1} \leq 1$$

This simply tells us sum of all the percentages of CPU time used per process has to be less than or equal to 100

B. Rate Monotonic (RM)

Rate monotonic analysis is a technique for assigning static priorities to periodic processes. As such, it is not a scheduler but a mechanism for governing the behavior of a preemptive priority scheduler. A conventional priority scheduler is used with this system, where the highest priority ready process will always get scheduled, preempting any lower priority processes.

A scheduler that is aware of rate monotonic scheduling would be provided with process timing parameters (period of execution) when the process is created and compute a suitable priority for the process. Most schedulers that support priority scheduling (e.g., Windows, Linux, Solaris, FreeBSD, NetBSD) do not perform rate monotonic analysis but only allow fixed priorities, so it is up to the user to assign proper priority levels for all real-time processes on the system. To do this properly, the user must be aware of all the real-time processes that will be running at any given time and each process' frequency of execution (1/T, where T is the period). To determine whether all scheduled processes can have their real-time demands met, the system has to also know each process' compute needs per period (C) and check that the following condition holds:

$$\sum_{i=1}^{n} \frac{C_i}{P_1} < \ln 2$$

III. Soft Real-Time Scheduling

IV. Results

V. Related Work

VI. Conclusion

References

[1] J. Ahmed and C. Chakrabarti, *A dynamic task scheduling algorithm for battery powered DVS systems*, 2004 IEEE International Symposium on Circuits and Systems, Vancouver, BC, 2004. doi: 10.1109/ISCAS.2004.1329396

[2] N. Audsley, A. Burns, R. Davis, K. Tindell and A. Wellings, *Real-time system scheduling*, 1995 Predictably Dependable Computing Systems, Springer Berlin Heidelberg, pp. 41-52.

[3] A. Chandrakasan, S. Sheng, and R.W. Brodesen. *Low-power CMOS digital design*. IEEE Journal of Solid-State Circuits, 27:473-484, April. 1992.

[4] R. Davis and A. Burns, *A survey of hard real-time scheduling for multiprocessor systems*, 2011 ACM computing surveys (CSUR), 43(4), 35.

[5] N. Guan, W. Yi, Z. Gu, Q. Deng and G. Yu, *New schedulability test conditions for non-preemptive scheduling on multiprocessor platforms*, 2008 Real-Time Systems Symposium, 2008. pp. 137-146. IEEE.

[6] C. Hung, J. Chen and T. Kuo, *Energy-Efficient Real-Time Task Scheduling for a DVS System with a Non-DVS Processing Element*, Real-Time Systems Symposium, 27th IEEE International, Rio de Janeiro, Brazil, 2006. doi: 10.1109/RTSS.2006.22

[7] D.Moren. *The next big thing in tech ought to be better batteries*. Macworld. August 19, 2016. Available at http://www.macworld.com/article/3109692/hardware/the-next-big-thing-in-tech-ought-to-be-better-batteries.html

[8] Y. W. Kwon and E. Tilevich, *Reducing the Energy Consumption of Mobile Applications Behind the Scenes*, 2013 IEEE International Conference on Software Maintenance, Eindhoven, 2013, pp. 170-179. doi: 10.1109/ICSM.2013.28

[9] W.-S. Liu. *Real-Time Systems*. Prentice Hall, Englewood Cliffs, NJ, June 2000.

[10] J. Luo and N.K. Jha, *Power-conscious Joint Scheduling of Periodic Task Graphs and Aperiodic Tasks in Distributed Real-time Embedded Systems*, IEEE/ACM International Conference on Computer Aided Design, San Jose, CA, 2000. doi: 10.1109/ICCAD.2000.896498

[11] R. Mall. *Real-Time Systems: Theory and practice*. Pearson Education India, 2009.

[12] H. Qian and D. Andresen, *An energy-saving task scheduler for mobile devices*. 2015 IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS), Las Vegas, NV, 2015, pp. 423-430. doi: 10.1109/ICIS.2015.7166631

[13] V. Rao, N. Navet and G. Singhal, *Battery aware dynamic scheduling for periodic task graphs*. 2006 Proceedings 20th IEEE International Parallel & Distributed Processing Symposium, Rhodes Island, 2006. doi: 10.1109/IPDPS.2006.1639403

[14] H. Takada and K. Sakamura, *Real-time synchronization protocols with abortable critical sections*. 1994 Proceedings of 1st International Workshop on Real-time Computing Systems & Application, pp. 48-52

[15] C. Tianzhou, H. Jiangwei, X. Lingxiang and W. Xinliang, *Balance the battery life and real-time issues for portable real-time embedded system by applying DVS with battery model*, 34th Annual Conference of IEEE, Florida Hotel & Conference Center, FL, 2008. doi: 10.1109/IECON.2008.4758018

[16] M. Weiser, B. Welch, A. Demers, and S. Shenker. *Scheduling for reduced CPU energy*. In Proc. of Symposium on Operating Systems Design and Implementation. Nov. 1994.

[17] C. Wilke, S. Richly, S. Götz, C. Piechnick and U. Aßmann, *Energy Consumption and Efficiency in Mobile Applications: A User Feedback Study*, 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, Beijing, 2013, pp. 134-141. doi: 10.1109/GreenCom-iThings-CPSCom.2013.45

[18] R. Ravishankar, V. Sarma and R. Daler N, *Battery modeling for energy aware system design*, Computer, IEEE Xplore Digital Library, Volume 36, Pages 77-87, 2003

[19] W. Yuan and K. Nahrstedt, *Energy-efficient soft real-time CPU scheduling for mobile multimedia systems*, ACM SIGOPS Operating Systems Review, Volume 37, Pages 149-163, 2003