EAI Endorsed Transactions on Collaborative Computing

Research Article EAI.EU

A Power Optimized Method for Mode Switching in Android Systems

Bo Chen[1,*], Xiaofan Shen[2]

[1]School of Software Engineering, University of Science and Technology of China, Hefei, Anhui
[2]Electrical and Electronic Engineering, Xi'an Jiaotong-Liverpool University(XJTLU), Suzhou, Jiangsu

Abstract

How the Suspend/Resume mechanism of smartphone influences the power consumption is examined in the dissertation. Specifically, various unimportant and not so urgent network packets keep awakening the operating system (OS) at the time it is under suspend mode, and switch it from suspend to resume mode continually, which results in more power consumption. Accordingly, an innovative optimization technique was suggested in this paper in order that the awakening of OS can be postponed and the lasting hour of suspend mode can be lengthened to decrease power consumption. Some experiments are also carried out, with the result data suggesting that such technique is an effective way to reduce power consumption by greater than 7.63%. It proves that this technique is workable.

Received on 08 May 2019; accepted on 26 July 2019; published on 06 August 2019

Keywords: Android, Power Save Mode, Suspend/Resume Mode

doi:10.4108/eai.9-10-2017.159797

1. Introduction

Nowadays, as mobile network communications develop, people cannot live without smartphone. As battery-powered personal mobile devices grow mature, low consumption of power is more popular and accepted in smartphone designing. There is an increasing demand for low power consumption, attracting extensive attention from research workers and experts from different fields[1][2][3]. In the opinion of power management[4][5], power consumption is under control by two main techniques: Dynamic Voltage and Frequency Scaling (DVFS), and Suspend/Resume (or sleep/active) technique. DVFS technique is a hardware-based and power-efficient mechanism which makes dynamic adjustment in processor implementation voltage to decrease the power consumption. In Suspend/Resume (or Sleep/Active) mechanism, CPU stays in different states of low power consumption without arranged system activities. Take android OS as an example. For lower power consumption, Suspend/Resume method enables the suspend of the kernel of android, and consequently every part (including DSP/bluetooth/Radio) is paused at the same time.

Normally, the smartphone will be switched to resume mode due to certain outer activities like background application message, pushnews, etc., when the android OS has stayed in suspend mode for some time. Based on some experimental facts, it was observed that the transmission of a few trivial network packets are able to activate the system continually and the smartphone will be turned to resume mode. In this way, for receiving these packets, the WiFi component will be switched to resume mode, so more power is consumed. As a matter of fact, it is feasible to postpone the awaking of the OS and delay the transmission of such less urgent packets.

In this paper, an optimization method was proposed to postpone the system activation when the smartphone is in suspend mode. It can postpone the awakening of the system intermittently and can deal with all the paused and postponed transmission of network packets for one time. Therefore, the power consumption expended by smartphone can be decreased largely.

*Corresponding author. Email: chenbo2008@ustc.edu.cn

This is an extended version of a paper presented at CollaborateCom 2018 - 14th EAI International Conference on Collaborative Computing: Networking, Applications and Worksharing

2. Suspend & Resume Mechanism

Suspend & Resume method [6] is a big function provided by Android kernel to lower the power consumption expended by mobile devices. Under such method, the system will be switched to suspend mode in a rapid manner without running any tasks to reduce power consumption.

In the other respect, under suspend mode, the system will be activated to deal with relevant tasks by the transmission of network packets of WiFi component. Under such situation, the system will be activated continually by simple and not important events/network messages from the outside, to the suspend mode finally, which will consume more power.

2.1. Impact of network transmission on power consumption

From the results of many experiments, it can be found that the transmission of network packets impacts the power consumption greatly under the suspend mode of Android OS. It is obvious from the Fig.1 that, the lower part indicates the network packets transmission, and the other part indicates the mode switching resulted from the packets transmission of Android OS. As the figure shows, at the time the network packets arrived at Access Point(AP), the system was activated by the WiFi module to receive the packets (which is marked by the shape of oval in Fig.1), and returned to suspend mode after some time.

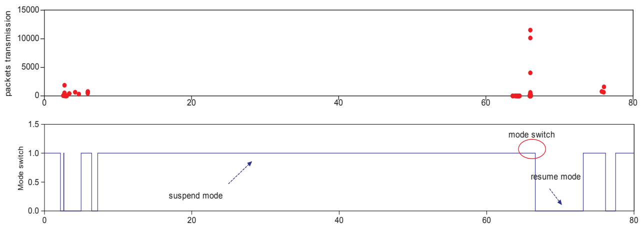

Figure 1. Impact of network transmission behavior on suspend/resume mode switching

We can observe that power consumption is strongly influenced by the frequent transmission of trivial network packets, according to which an optimization method was put forward to postpone the activation of all hardware components, to reduce the power expended by WiFi component, and in the meanwhile, to decrease the power expended by the whole device.

2.2. IEEE PS-POLL mechanism

When it comes to WiFi component, attention was paid to if the original network communication protocol would be influenced when the transmission of packets was postponed.

As to WiFi network, from IEEE 802.11 protocol, to reduce power consumption, part of the transceiver devices will be switched off for a certain period if Power Save mode is chosen by a client station (smartphone). Regarding communication behaviors, smartphone will receive the beacon frame from Access Point(AP) in a periodic manner which shows any data that has reached AP within the frame if PS mode is set on WiFi component. When no packets arrive, the *TIM* field is set to zero (*TIM=0*). It will be set to 1 (*TIM=1*) if any packets have been received. After that, AP will receive PS-Poll frame from smartphone to indicate its activation for data receiving, which can be seen in Fig.2.

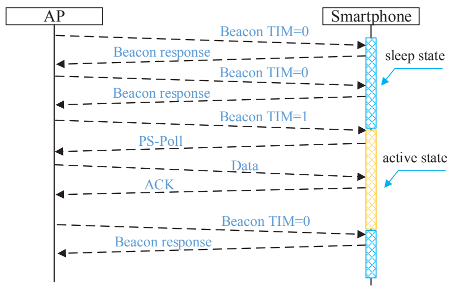

Figure 2. The behavior of router retrieves network packets

In fact, the PS-POLL (which indicates that the OS has been waken up and ready to receive packets) beacon will also be delayed when being sent to AP through the delay of the activation of the android OS. The figure shows that the original network protocol will not be influenced when the sending of the PS-POLL frame is delayed.

2.3. Related Works

During the last ten years, there have been many investigations into the field of lower power consumption of smartphone. For instance, Niranjan [6] presented a study of the power consumption features of WiFi, 3G, and GSM by means of classical measurement, with a result of a model for the energy consumed by network activity regarding different techniques. Then TailEnder was designed which was a protocol to lower the power consumption by commonly seen mobile devices. Reviewing one that belongs to the most integrated WiFi power models[7], Swetank made a re-assessment on the smartphones of the latest generation with both 802.11g and 802.11n NICs. Results came that they were still valid on certain component and network kinds, in spite of the fact that their parameters presented a different picture from those recorded in the original paper.

Another interesting works aimed at optimizing network protocol to lower power consumption [8][9][10][11]. For example, in paper [8], they studied the influence of network protocol on the energy expended. Concretely, the TCP and UDP protocols were checked in both on-and-off-power conditions of the screen, as packets were being transmitted to an AP by the smartphone. These results were attained with 802.11n/ac wireless NICs. It is also a valuable guidance to us.

3. Delay Wake-Up Mechanism

Based on the discussion above, the expenditure of energy will be influenced by network packets that are less urgent. Consequently, an optimization method was put forward in the paper to postpone the system activation as well as postpone the switch of the Android OS to resume mode. From Fig.3 the postponing of three beacon intervals can be found in the response beacon.

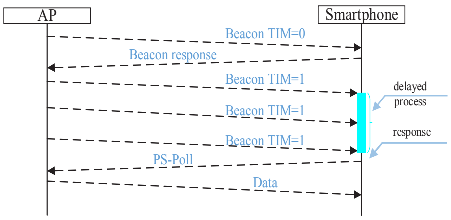

Figure 3. The dealy transmission behavior of router retrieves network packets based on PS-Poll mechanism

3.1. Suspend & Resume operation of Android OS

Under suspend mode, a large loop program will be conducted by the kernel awaiting the interrupt event (WiFi interrupt) or signal (screen unlock signal), etc. from the outside. With the coming of such an event, every sub-component will be activated by the system, and relevant interrupt handler will also be activated to deal with it. It can be seen below how the resume mode is switched to suspend mode (Fig.4):

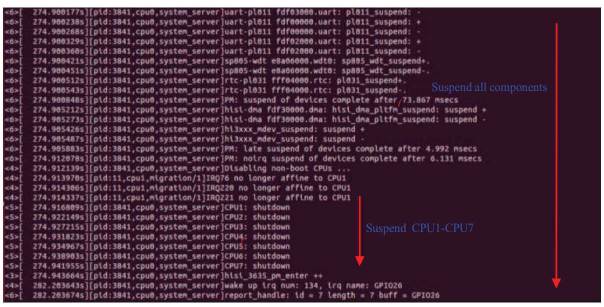

Figure 4. Switch operation flow of Suspend mode

It can be seen from Fig.4 that nearly every hardware component is paused successively at the beginning. As observed the last few columns of execution are to switch off the CPU (not including CPU0), and a large loop task will be conducted to deal with such event/signal etc. It is how the android OS is paused.

Analogously, with the arrival of an interrupt event from the outside (*irq name:GPIO26*, actually it is a WiFi interrupt which means AP has received packets), it will activate the system. The complete pause period is approximately 7.27 (282.20-274.94) seconds.

It can be seen in Fig.5 that every component will be awakened after that in the system (first the CPU1-CPU7 and then the rest hardware components).

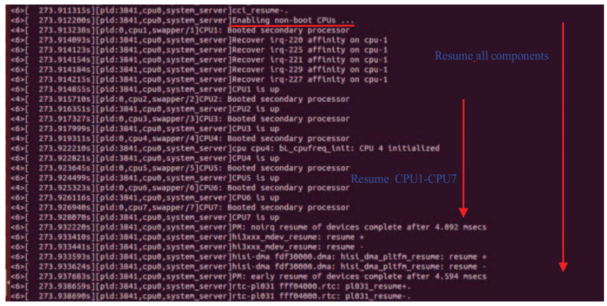

Figure 5. Switch operation flow of Resume mode

The mechanism will postpone the activation of the system at the detection of outer interrupts (an example is the interrupt caused by the WiFi component), so as to lengthen the time Android OS stays in suspend mode.

3.2. Resume operation of Android OS API

We have demonstrated where the function call goes after an outer interrupt event was detected in the system. It can be seen from Fig.6 that when the interrupt reached, the interrupt handler (Dhd_dpc_thread()) was called first (see Fig.6). Eventually, netif_rx was called to submit the network packets to the upper network protocol layer by the driver.

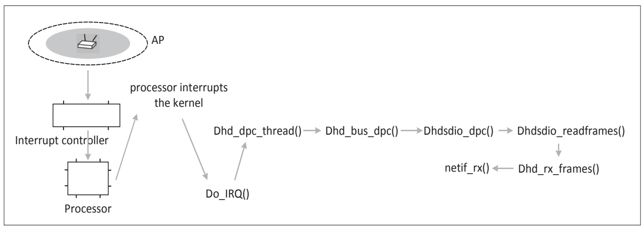

Figure 6. Trace of API for Resume operation

3.3. Implementation of delayed wake-up operation

The optimization approach postpones the response to the WiFi interrupt, and thus delay the transmission of PS-POLL frame as well as the restart of the CPU, etc., for the purpose of postponing the switch from suspend mode to resume mode.

Below shows how to set up the WiFi interrupt processor:

Algorithm 1 /bcm/wifi/driver/bcmdhd/dhd_linux.c

```
Dhd_dpc_thread(void *data)

unsigned long delay = jiffies + 3*HZ;
while 1 do        //Run until signal received
    if !binary_sema_down(tsk) then        //When external event is received
        if responese_number > 0 then
            if dhd->pub.sus ==1 then
                my_flag_s = 1;
                ssleep(3);        //Delay 3 seconds
                my_flag = 1;        // Setting the flag of my_flag
                wake_up_interruptible(&my_queue);    //wake up the queue
            end if
        end if
    end if
end while
```

When the interrupt signal reaches, there is a period before the system is activated. To realize the delay, we use the function of wait_queue() method, which is provided by Linux kernel. During such process, CPUs will be restarted immediately after the period postponed runs out (see below).

Algorithm 2 /kernel/net/core/dev.c

```
void __ref enable_nonboot_cpus(void)

    int cpu, error;
    int my_f, my_f_s;
    my_f = get_my_flag(0);        //Get the value of the global variable my_f
    my_s=get_my_flag_s(0);        //Get the value of the global variable my_flag_s
if my_f_s == 1 then
    if ( thenwait_event_interruptible(my_queue, my_flag!= 0 ))        //When 3 seconds expires, the variable of my_flag will be set to 1
        return 0;
    end if
end if
    my_flag = 0;        /Reset the variable my_f
    my_flag_s = 0;        /Reset the variable my_flag_s
    printk(KERN_INFO "Enabling non-boot CPUs ...");
    arch_enable_nonboot_cpus_begin();
```

3.4. Formal description of optimization Approach

We describe the dynamic behavior of optimization approach by automata[12], which is shown in Fig.7. We can see from the figure that the WiFi component works for the transmission of network packets, after a period (idle_time > threadhold_1) when there is no data transmission, then the WiFi component will switch from Resume state to Suspend state. When AP receives data which will be sent to the smartphone, the WiFi component will receive an interrupt to wake up the OS. Based on our optimization approach, WiFi component will switch from Suspend Mode to Delayed Mode. Finally, after *dealyed_time* seconds, WiFi will switch to Resume mode and is ready to receive network packets.

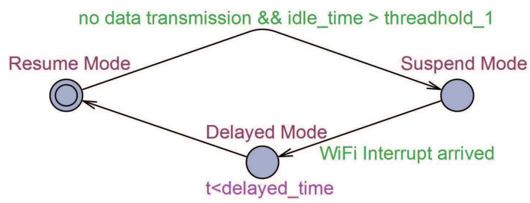

Figure 7. The automata model of optimization approach

3.5. Delay time Setting

User experience should be considered when using optimization approach, for the purpose of getting a proper delay time. If it is comparatively large, power efficiency can be better improved at the expense of worse user experience. But the corresponding, network packets might even be lost, or even re-transmission might be needed for several times if the delay time is set to be too large. But how to get a proper postponing time? We have conducted many experiments to analyze the communication behaviors of various applications like WeChat, QQ, Microblog, especially the interval between the interrupts caused by WiFi components (as shown in the bottom line of Table 1). The statistics are assessed and analyzed as below:

Table 1. Time distribution of the interval between the packet's arrival

APP	1-6(s)	6-10(s)	>10(s)
MicroBlog	30.45%	8.69%	60.86%
Wechat	30.59%	15.78%	53.63%
QQ	39.39%	12.13%	48.48%
QQ+Webchat+MicroBlog	56.53%	15.21%	28.26%

The above table shows that WiFi components will arrive within 1-6s about 56.53% in all cases, and causes the android OS to switch to resume mode, when related

WeChat+QQ+MicroBlog are all in operation. But when it comes to the single behavior of the MicroBlog app (see the top line of the table), over 30.45% will switch the Android OS to resume mode after staying in suspension less than 6 seconds.

To balance the performance with the experience and experience of user, the postponing period of separately1/3/5/10 seconds can always obtain the optimal performance.

4. Evaluation

The proposed approach was implemented on the Huawei-P8 mobile devices with the rooted Android 6.0 OS, with the WiFi connector of BCM4334 chipset.

4.1. Power Model of Smartphone

We obtain the file of power_profile.xml via Apktool from the kernel category of /framwork-res/res/xml/. in the format of < item name="wifi.on">0.06</item>, <item name="wifi.scan "> 100 </item>, etc.. In this way, we can measure the power expenditure through the hardware parameter when the operation system is under both modes.

The suspend mode of the operation system leads to the pause of almost all hardware components, not including one CPU core, which demonstrates that improved energy efficiency is not limited to WiFi module. It is workable for the rest components as well. The following equation can be adopted to measure the energy consumption.

The energy consumption can be measured for all the components separately when the expenditure of the whole smartphone is separated into 5 parts.

$$E_{total} = E_{cpu} + E_{bluetooth} + E_{WiFi} + E_{Dsp} + E_{Radio} \quad (1)$$

1. CPU parameters relevant to energy consumption

Under suspend mode, merely one CPU core implements tasks necessary for the kernel. For instance, periodic activities are conducted to activate the system. It will also check whether there are outer interrupt events. Consequently, the energy expenditure of both modes should be measured respectively. Below is the rough calculation for the energy expenditure of CPU:

$$\begin{cases} E_{cpu\_resume} = Time_{cpu\_resume} \times Power_{cpu\_resume} \times 8 \\ E_{cpu\_suspend} = Time_{cpu\_suspend} \times Power_{cpu\_suspend} \end{cases} \quad (2)$$

2. WiFi parameters relevant to energy consumption

Regarding WiFi component, we measure the consumption of power by means of <item name= "wifi.on">0.06</item> under suspend mode, and measure the energy parameter of WiFi by means of <item name="wifi.active">97</item>.

In contrast, below are the energy consumption for all components (see table 2).

Table 2. Different Power Consumption under suspend/resume mode

	CPU	WiFi	DSP	Bluetooth	Radio
Suspend(mAh)	348.3	97	91	116	117
Resume(mAh)	3	0.06	0	2.8	37.5

4.2. Experimental Setting and Experimental Result Analysis

At the end, we conducted many experiments to compare between the impact of postponing time on performance and that on loss of packets, to check whether the proposed approach is useful.

Preparations include the development of a C/S-architecture-based network application, the gathering of time used in sending and receiving the packets separately, and the measurement of packet postponing and loss of packets. The former is the end-to-end postponing between specified packets within the flow where there is loss of packets, and the latter is that some packets cannot reach their designated place during the transfer. Below are the results (100 times of packets transfer were conducted at random intervals).

The postponing period was made 1s, 3s, 5s and 10s separately. After that is the collection of the time used in sending and receiving packets, and the lasting hours of both modes. The assessment of the method put forward was made in the following three facets: (a) the lasting time of both modes; (b) the postponing of packet transmission; (c) the improvement of performance. Below are the results in details:

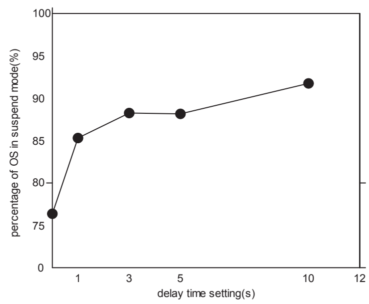

Figure 8. Percentage of OS spend in suspend mode

For the first facet, as indicated in Fig.8, if the postponing period was not set, the android OS will stay in suspend mode for almost 76.38% of time during the overall communication process. If the delay time was set to 1s, it occupied 85.31% of the total; if the period was set to 10s, the percentage increased to 91.75%.

Regarding the second facet, as can be found in Fig.9, if the delay time was not set, no delay would be caused. If the delay time was set to 1s, there would be a 0.12s delay in transmitting packets. If the time was set to 10s, the delay climbed to nearly 7.86s.

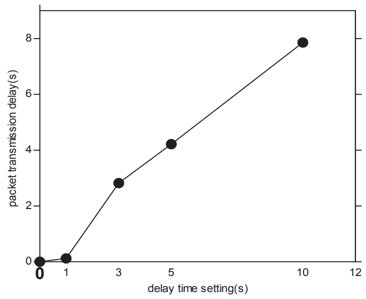

Figure 9. Packet transmission delay after using new mechanism

As with the third facet, experiments were conducted on every single application and the combination of the three (see Fig.10).

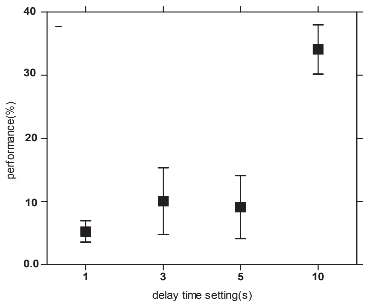

Figure 10. Percentage of power saving after using proposed mechanism

If the delay time was set to 1s, the power was reduced by lower than 6.92%; if the delay time was set to 3s, the power reducing rate was between 4.72% and 15.32%; if the period was 10s, the rate fell between 30.19% and 37.96%.

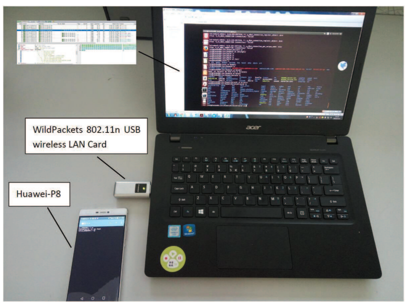

Figure 11. Picture of the experimental platform

The experimental platform is shown as Fig.11, and we use wildpackets 802.11n usb wireless LAN card as wireless Access Point(AP). Meanwhile, we obtaining the statistical data of the network communication behaviors by the Packet sniffing tool.

5. Conclusion and Prospect

When android OS is in suspend mode, non-urgent external network packets transmission will cause the system to switch from suspend mode to resume mode, and thus leads to more energy consumption. For this reason, an optimization mechanism was proposed in this study to delay receiving WiFi packets, and prolong the duration of WiFi component in suspend mode. According to the experimental data, it can be known that the proposed mechanism can effectively reduce the power consumption with high feasibility.

Acknowledgments. This work was supported by the Suzhou Scientific Research Program (No.: SYG201731), and the National Natural Science Foundation of China (No.:61772482, 61303206).

References

[1] Ding N, Hu Y C. GfxDoctor: A Holistic Graphics Energy Profiler for Mobile Devices[C] //Proceedings of the Twelfth European Conference on Computer Systems. ACM, 2017: 359-373

[2] Chen B, Li X, Zhou X, et al. Towards Energy Optimization Based on Delay-Sensitive Traffic for WiFi Network[C]// Ubiquitous Intelligence and Computing, 2014 IEEE, Intl Conf on and IEEE, Intl Conf on and Autonomic and Trusted Computing, and IEEE, Intl Conf on Scalable Computing and Communications and ITS Associated Workshops. IEEE, 2015:252-259

[3] Chen B, Li X, Zhou X. PowerSensor: A method for power optimization of smartphone through sensing wakelock application[C]//2017 9th International Conference on Wireless Communications and Signal Processing (WCSP). IEEE, 2017: 1-6.

[4] Chen B, et al. A low-power transmission approach of WiFi network in smartphone. (2017): 3-5.

[5] Li H, Chen L. RSSI-aware energy saving for large file downloading on smartphones[J]. IEEE Embedded Systems Letters, 2015, 7(2): 63-66.

[6] Balasubramanian N, Balasubramanian A, Venkataramani A. Energy consumption in mobile phones: a measurement study and implications for network applications[C] //Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement. ACM, 2009: 280-293.

[7] Saha S K, Malik P, Dharmeswaran S, et al. Revisiting 802.11 power consumption modeling in smartphones[C]//2016 IEEE 17th International Symposium on. IEEE, 2016: 1-10.

[8] P. Serrano, A. Garcia-Saavedra, A. Banchs, G. Bianchi, and A. Azcorra, Per-frame energy consumption anatomy of 802.11 devices and its implication on modeling and design, IEEE/ACM Transactions on Networking (ToN), vol. 23, no. 4, pp. 1243ÍC1256, 2014.

[9] Andrew Rice and Simon Hay. Decomposing power measurements for mobile devices. In Eighth Annual IEEE

International Conference on Pervasive Computing and Communications (PerCom), Page(s):70-78, Mar 2010.

[10] Kim, S., Kim, H.. An event-driven power management scheme for mobile consumer electronics. Consumer Electronics, IEEE Transactions on, Page(s): 259- 266 ,59(1),2013.

[11] S. K. Saha, P. Deshpande, P. P. Inamdar, R. K. Sheshadri, and D. Koutsonikolas, Power-throughput tradeoffs of 802.11n/ac in smartphones, in Proc. of IEEE INFOCOM, 2015.

[12] Chen B, Li X, Zhou X. Model checking of MARTE/CCSL time behaviors using timed I/O automata[J]. Journal of Systems Architecture, 2018, 88: 120-125.