Delayed Wake-Up Mechanism Under Suspend Mode of Smartphone

Bo Chen(✉), Xi Li, Xuehai Zhou, and Zongwei Zhu

University of Science and Technology of China, Hefei 215123, China
chenbo2008@ustc.edu.cn

Abstract. In this paper, the impact of Suspend/Resume mechanism on the power consumption of smartphone is investigated. When the operating system (OS) is in suspend mode, many trivial (less urgent) network packets will always wake up system, making OS frequently switch from suspend mode to resume mode, thus causing a high power consumption. Based on this observation, an novel optimization mechanism was proposed to delay wake-up system and prolong the duration of suspend mode so as to reduce power consumption. This novel method not only reduces the power consumption of WiFi components, but also reduces the power consumption of the total components. To verify the effectiveness of the novel mechanisms, we implemented such proposed mechanism on Huawei P8 platform, and carried out relevant experiments. The results showed that the proposed mechanism can effectively decrease power consumption by more than 7.63%, which indicates the feasibility of the proposed mechanism.

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

1 Introduction

With the development of mobile network communications, smartphone has become an important part of our daily life. With the successful development of battery-based personal mobile devices, low power consumption has become a major trend in the development of smartphone. The problem of high power consumption is becoming more and more serious, which has been widely concerned by researchers and scholars in various fields [1–3]. From the point of view of power management, there are mainly two mechanisms controlling the power consumption, i.e. Dynamic Voltage and Frequency Scaling (DVFS) and Suspend/Resume (or sleep/active) method. DVFS method is a low-energy consumption mechanism supported by hardware, which dynamically adjusts the voltage for processor implementation and decreases the system power consumption by extending operation time of the tasks. Suspend/Resume (or Sleep/Active) method brings the CPU to various low-power states when there are no activities scheduled in the systems. For instance on android OS, to reduce the power consumption, Suspend/Resume mechanism can make the android kernel suspended, and thus all components (DSP/bluetooth/Radio, etc.) are suspended simultaneously.

In general, after the android OS has stayed in suspend mode for a period of time, some external events (Timer/signal/hardware interrupt. etc.) will make the smartphone switched to resume mode. A number of experiments show that small amounts of trivial

H. Gao et al. (Eds.): CollaborateCom 2018, LNICST 268, pp. 346–355, 2019.
https://doi.org/10.1007/978-3-030-12981-1_24

network packets transmission can frequently 'wake-up' the system and make the smartphone switched to resume mode. Therefore the WiFi component will be woken up to resume mode to receive the network packets, and thus more power consumption will be consumed. In fact, a delay processing of trivial network packets transmission is possible.

In this paper, an optimization mechanism was proposed to realize delayed "wake up" mechanism of smartphone under the suspend mode. Such mechanism will delay system resuming in a certain interval of time, and process all the buffered network packets transmission at one time. In this way, it can effectively reduce the power consumption of smartphone.

2 Suspend & Resume Mechanism

Suspend & Resume mechanism [4] is a huge function provided by Linux kernel to decrease the power consumption of mobile devices. Suspend & Resume mechanism will try to put the system into a sleep mode as soon as possible while executing no task for saving more power consumption.

However, WiFi component or CPU will hold the Wake lock of the 'Suspend' mode, then the system will not be suspended. On the other hand, when system is in the suspend mode, the packets transmission of WiFi network or the interruption of Timer will resume the system to handle the relevant operations. In this case, the trivial external events will always wake up the system and eventually switch the smartphone to Suspend mode again. Such repeated process will consume more power consumption.

2.1 Impact of Network Transmission on Power Consumption

A large number of experiments have shown that the transmission of network packets has a great influence on the power consumption when Android OS is in suspend mode. As shown in the Fig. 1, the lower part represents the packets transmission in suspend mode, while the upper part represents the switching of mode caused by packets transmission of Android OS. It can be seen that, when the network packets arrived at Access Point (AP), the WiFi component woke up the system to receive the network packets, and after a period of time, the system was switched back to suspend mode again.

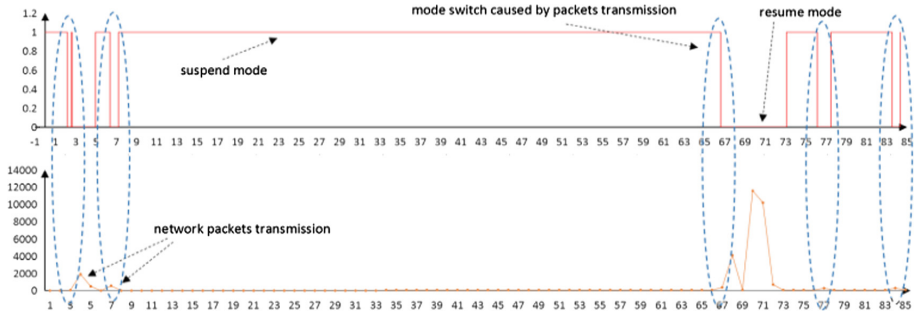

Fig. 1. Impact of network transmission behavior on suspend/resume mode switching

It can be known that frequent transmission of less urgent packets has a great impact on power consumption. Based on this, an optimization mechanism was proposed to decrease the power consumption of WiFi component while delaying the resuming of all hardware components, also in order to decrease the power consumption of total device.

2.2 IEEE PS-POLL Mechanism

From the aspect of WiFi component, focus was paid on whether delayed transmission of packets could affect the original network protocol.

For WiFi network, when a client station (smartphone) is set to Power Save mode, some of the transceiver equipment will be shut down to conserve power. Considering the communication behavior, when the WiFi components are in PS mode, Access Point (AP) will periodically send the beacon frame to smartphone to indicate in the frame whether there are any data reaching the AP. If there are no packets reaching AP, the TIM field is set to 0 (TIM = 0). When some packets have reached AP, the TIM is set to 1. Then, smartphone will send PS-Poll frame [5] to AP, which indicates that the smartphone has been woken up to receive data, as shown in Fig. 2.

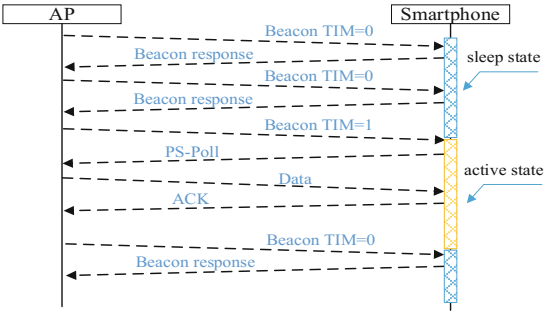

Fig. 2. The behavior of router retrieves network packets

It has been proved that the proposed mechanism will delay sending the PS-POLL frame to AP without affecting the original network protocol.

2.3 Related Works

In the past decade, a lot of researches have been conducted on reducing the power consumption and increasing the energy efficiency. For example, Niranjan [6] conducted a classical measurement study of the energy consumption characteristics of 3G, GSM, and WiFi, and developed energy consumption model for each technology. Based on this model, they developed TailEnder, a protocol that reduces energy consumption of common mobile applications. By revisiting one of the most comprehensive WiFi power models [7], Swetank re-evaluated the model for the current generation of smartphones equipped with both 802.11g and 802.11n NICs, and found that such model remained valid for a wide range of devices and network types, although the model parameters exhibited different trends compared with the ones reported in the original paper.

Another interesting work is to optimize network protocol and thus reduce energy consumption [8–10]. In [11], the impact of network protocol on the power consumption was studied [11]. The author examined the TCP and UDP protocols with screen on and off, while the smartphones forwarded the packets to an Access Point. The findings were obtained using 802.11n/ac wireless cards, which is of great guidance value.

3 Delay Wake-Up Mechanism

According to the above content, non-urgent network packets will increase the power consumption. Therefore, an optimized mechanism was proposed in this work to delay the wake-up of the system and make the Android OS late switched to resume mode. It can be seen from the Fig. 3 that the response beacon has been delayed by the 3 beacon intervals.

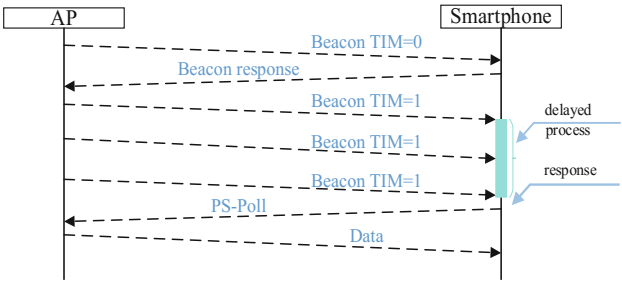

Fig. 3. The delay transmission behavior of router retrieves network packets based on PS-Poll mechanism

3.1 Suspend & Resume Operation of Android OS

When the system is in suspend mode, kernel will execute a big loop program to wait for the external interrupt events or signals. When an external event is arrived, the system will wake up all the sub-components and call the corresponding interrupt handlers to handle the event. The switching operation from resume mode to suspend mode is as follows (Fig. 4):

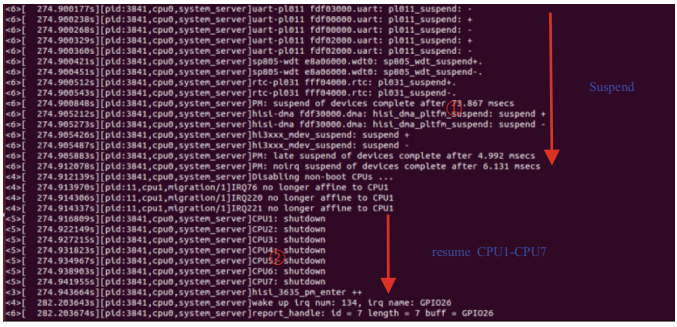

Fig. 4. Switch operation flow of Suspend mode

As shown in Fig. 3, almost all hardware components are suspended in the first step. It can be seen that the last operation is to shutdown all CPU except CPU0, which will execute a big loop task for handling the external events/signals. That is the suspension flow of the android OS.

Similarly, when an external interrupt event is received (*irq name:GPIO26, actually it is an WiFi interrupt which means AP has received packets*), the system will be woken up. The total suspend time is about 7.27 (*282.20–274.94*) s.

After that, the system will resume all components (first the CPU1–CPU7 and then the other hardware components), as shown in Fig. 5.

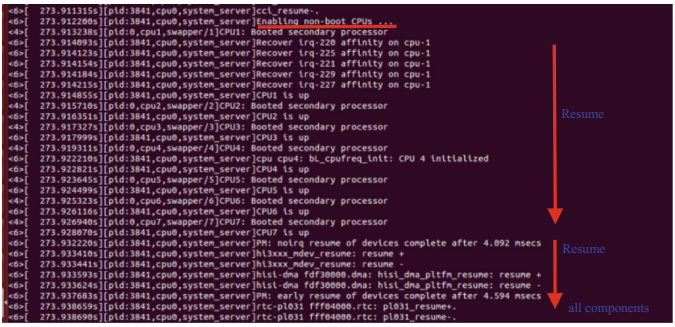

Fig. 5. Switch operation flow of Resume mode

To prolong the duration of suspend mode, the proposed mechanism will delay the wake-up of the system when the external interrupts are detected (for instance, the interrupt generated by the WiFi component, which is quite rightly our concern).

3.2 Resume Operation of Android OS API

The trace of the function call is investigated when the system receives an external interrupt event. As shown in Fig. 6, after the WiFi component receives the interrupt, and the interrupt handler function (*Dhd_dpc_thread()*) is invoked. Finally, the driver calls *netif_rx* to invoke the handler for the upper network protocol layer.

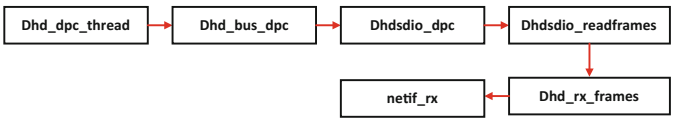

Fig. 6. Trace of API for Resume operation

3.3 Implementation of Delayed Wake-up Operation

To accomplish the delayed switching from suspend mode to resume mode, the proposed mechanism delays the response to the WiFi interrupt event, which leads to a delay in sending the PS-POLL frame and rebooting the CPU.

The implementation process of WiFi interrupt handler is as follows:

```
/kernel/drivers//bcm/wifi/driver/bcmdhd/dhd_linux.c

/*Run until signal received*/
   While(1) {
     if (!binary_sema_down(tsk))  {
    if (responese_number > 0)
       if(dhd->pub.sus ==1)
```

After WiFi network receives interrupt signals, the wake-up of system will be delayed for a while. The wait_ queue() mechanism is used to implement such delay operation. This mechanism makes processes wait for the occurrence of a particular event. In the implementation process, when the delay time is up, CPUs will be rebooted firstly, as shown as below.

```
/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);
      my_s=get_my_flag_s(0);
      if(my_f_s == 1);
      .....
}
```

3.4 Delay Time Setting

To set the delay time more reasonably, we need to balance between the performance and the user experience. If the delay time is set to be relatively large, a lower power optimization effect can be realized, but the user's interactive experience will be negatively affected. A too long delay time will lead to the loss of network packets or even multiple re-transmission. To obtain an appropriate delay time, we evaluated the communication behaviors of the different apps such as *WeChat*, *QQ*, *Microblog*, and analyzed the interval between the packet's arrival of the combination of apps (see the last line in the table 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+Wechat+MicroBlog	56.53%	15.21%	28.26%

From the Table 1, it can be seen that for *Wechat* alone, more than 53.63% of packets will arrive within 10 s. However, for the composite behaviors of the last line (QQ+Wechat+MicroBlog), more than 56.53% of the network packets will arrive within less than 6 s. Thus, to balance between the performance and user experience, the delay time is respectively set to be 1/3/5/10 s.

4 Evaluation

The proposed mechanism is implemented on the Huawei-P8 platform on which the rooted Android 6.0 OS has been installed, and BCM4334 chipset is used as WiFi connector. In this chapter, the power consumption is calculated by the power model of smartphone, and the experimental setting and result analysis are presented.

4.1 Power Model of Smartphone

Apktool tool is used to extract the *power_profile.xml* file from the kernel directory of / framwork-res/res/xml/. The format is *<item name = "wifi.on"> 0.06 </item>*, *<item name = "wifi.scan"> 100 </item>, etc.* Then, the power consumption can be calculated by such hardware parameter when Android OS is in suspend or resume mode.

Nearly all hardware components are suspended except one CPU core when android OS enters Suspend mode. This indicates that the power saving is not only for WiFi component, but also for the other components. The power consumption is calculated using the formula as below.

$$E_{total} = E_{cpu} + E_{Bluetooth} + E_{WiFi} + E_{Dsp} + E_{Radio}$$

By dividing the total power consumption of smartphone into five parts, it is possible to separately calculate the power consumption of each component.

- *CPU parameters relevant to power consumption*

When Android OS enters suspend mode, there is only one CPU core executing the operation which is the necessarily needed by kernel. For example, it will execute the periodic task to detect (external) events and wake-up the system. Thus, the power consumption under each working mode should be calculated separately. The rough formula for calculating power consumption of the CPU is as below:

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

- *WiFi parameters relevant to power consumption*

For WiFi component, when OS enters suspend mode, <*item name* = "*wifi.on*"> *0.06* </*item*> can be used to calculate the power consumption. When OS stays in resume mode, <*item name* = "*wifi.active*"> *97* </*item*> can be used to represent the WiFi power parameter.

As an analogy, the power consumption of each component is shown in Table 2.

Table 2. Time distribution of 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

Finally, to verify the effectiveness of the proposed mechanism, comparison experiment was carried out to study the influence of delay time on smartphone performance and packet loss.

A network application was designed based on the C/S architecture. The time for sending and receiving the network packets was collected, respectively. The packet delay (the end-to-end delay between selected packets in a flow with any lost packets) and the packet loss (the failure of transmitted packets to arrive at their destination) were calculated. The statistical results are shown as follow (the packet is sent 100 times at a random time interval).

The delay time was set to 1 s, 3 s, 5 s and 10 s, respectively. Then, the sending and receiving time of packets as well as the duration of suspend mode and resume mode were collected. The proposed mechanism was evaluated from mainly three aspects: (1) the duration of suspend mode and resume mode. (2) the packet transmission delays. (3) the performance gains.

For the first aspect, the statistical result is showed in Fig. 7. When no delay time was set, the duration of suspend mode accounted for 76.38% of total working time. When the delay time was 1 s, the duration of suspend mode accounted for 85.31% of total working time. When the delay time was 10 s, the duration of suspend mode accounted for 91.75% of total working time.

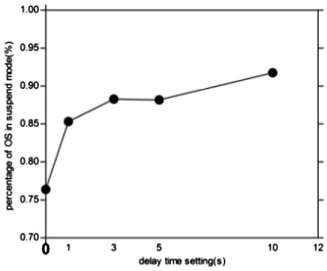

Fig. 7. Percentage of OS in suspend mode

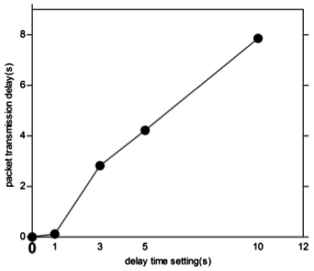

Fig. 8. Packet transmission delay after using new mechanism

For the second aspect, the statistical result is showed in Fig. 8. When no delay time was set, there was almost no delay. When the delay time was 1 s, the packet was delayed by 0.12 s. When the delay time was 10 s, the packet was delayed by about 7.86 s. This indicates the average time from end to end in the proposed mechanism is significantly larger than before.

For the third aspect, the performances of each APP alone and combination of APPs (QQ+Wechat+MicroBlog) were tested, and the statistical result is shown in Fig. 9. When the delay time was set to 1 s, the energy saving was less 6.92%. When the delay time was set to 3 s, the percentage of energy saving was between 4.72% to 15.32%. When the delay time was set to 10 s, the energy saving was between 30.19% to 37.96%.

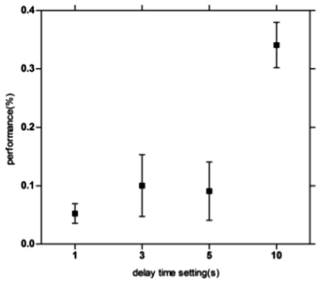

Fig. 9. Percentage of power saving after using proposed mechanism

5 Conclusion and Prospect

When android OS is in suspend mode, unexpected external network packet transmission will cause the system to switch from suspend mode to resume mode, and thus leading to more energy consumption. For this reason, an optimization mechanism was proposed in this study to delay receiving WiFi packet, and prolong the duration of suspend mode. According to the experiment data, it can be known that the proposed mechanism can effectively reduce the power consumption and has 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. In: Proceedings of the Twelfth European Conference on Computer Systems, pp. 359–373. ACM (2017)
2. Chen, B., Li, X., Zhou, X., et al.: Towards energy optimization based on delay-sensitive traffic for wifi network. In: 2014 IEEE International Conference on Ubiquitous Intelligence and Computing and IEEE International Conference on and Autonomic and Trusted Computing and IEEE International Conference on Scalable Computing and Communications and ITS Associated Workshops, pp. 252–259. IEEE (2015)
3. Li, H., Chen, L.: RSSI-aware energy saving for large file downloading on smartphones. IEEE Embedded Syst. Lett. **7**(2), 63–66 (2015)
4. Ding, N., Wagner, D., Chen, X., Hu, Y.C., Rice, A.: Characterizing and modeling the impact of wireless signal strength on smartphone battery drain. In: Proceedings of ACM SIGMETRICS (2013)
5. Swain, P., Chakraborty, S., Nandi, S., Bhaduri, P.: Throughput analysis of the IEEE 802.11 power save mode in single hop ad hoc networks. In: Proceedings of ICWN (2011)
6. Balasubramanian, N., Balasubramanian, A., Venkataramani, A.: Energy consumption in mobile phones: a measurement study and implications for network applications. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, pp. 280–293. ACM (2009)
7. Saha, S.K., Malik, P., Dharmeswaran, S., et al.: Revisiting 802.11 power consumption modeling in smartphones. In: IEEE International Symposium on A World of Wireless. IEEE (2016)
8. Serrano, P., Garcia-Saavedra, A., Banchs, A., Bianchi, G., Azcorra, A.: Per-frame energy consumption anatomy of 802.11 devices and its implication on modeling and design. IEEE/ACM Trans. Netw. (ToN) **23**(4), 1243–1256 (2014)
9. Rice, A., Hay, S.: Decomposing power measurements for mobile devices. In Eighth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 70–78, March 2010
10. Kim, S., Kim, H.: An event-driven power management scheme for mobile consumer electronics. IEEE Trans. Consumer Electron. **59**(1), 259–266 (2013)
11. Saha, S.K., Deshpande, P., Inamdar, P.P., Sheshadri, R.K., Koutsonikolas, D.: Power-throughput tradeoffs of 802.11n/ac in smartphones. In: Proceedings of IEEE INFOCOM (2015)