An Rssi-Based Wi-Fi Download Management Approach for Energy Saving on Smartphones

Lunde Chen, Huan Li*, *Member, IEEE*

***Abstract*—Smartphones are emerging as a particularly appealing platform for network applications, especially through the Wi-Fi interface. However, Wi-Fi entails considerable energy consumption and smartphones are bottlenecked by their battery capacity. Finding ways to reduce energy consumption of Wi-Fi is more than critical. In this paper, we address the problem of energy saving of Wi-Fi with regarding to Rssi. Through extensive experiments, we investigate how Rssi influences energy consumption. Based on our experimental findings, we propose a novel Rssi-based Wi-Fi download management approach for energy saving on smartphones which is simple but highly elaborate. Simulations prove that our approach is effective and can achieve more than 100% energy saving, both in a constantly moving scenario and in a stationary-moving scenario.**

***Index Terms*—Rssi, Energy saving, algorithm.**

I. Introduction

SMARTPHONES are becoming increasingly popular and have emerged as a particularly appealing platform for network applications, especially through the Wi-Fi interface. However, these light-weighted and easy-to-carry mobile devices are constrained by their limited battery capacity. Considering the scenario where people use their smartphones to download large files such as movies, documents and apk files, typically larger than 10 MB and taking several minutes or longer to download, one of the key challenges is how to perform the download in an energy-efficient way.

We investigate the relationship between the energy consumption and Rssi and conclude that Rssi has a major impact on the energy consumption. Downloading one file in weak Rssi environment can consume 8 times more energy than in strong Rssi. Therefore, an Rssi-based Wi-Fi download management scheme is needed to achieve good energy efficiency. The ideal is to download files where Wi-Fi signal is strong and stable. However, smartphones are naturally for mobile usage, resulting in a constantly changing Rssi environment. In the other hand, even when the smartphone is stationary, Rssi is not stable and demonstrates considerable fluctuations [1][2].

In this paper, we propose a finite state machine (FSM) based download management scheme to leverage the varying nature of Rssi and achieve energy savings, which is based on our extensive experiments to evaluate the relationship of Rssi and the energy consumption. We further conduct effectiveness evaluation of our scheme using simulations.

*Corresponding Author: H. Li, is with the School of Computer Science and Engineering, Beihang University, Beijing, China 100191, e-mail: lihuan@buaa.edu.cn.

L. Chen is with with the School of Computer Science and Engineering and Ecole Centrale de Pékin, Beihang University, Beijing, China 100191.

Manuscript received September 15, 2014.

II. Evaluation of Rssi and Energy Consumption

A. Experimental Setup for Power Measurement

The DUT (Device Under Test) is a Huawei 8950D running Android 4.0 with superuser access, supporting IEEE 802.11 n/b/g. It's equiped with a double-core 1.2GHz Snapdragon CPU and 768MB RAM. We use a Monsoon Power Monitor [9] for power measurement. The Monsoon Power Monitor supplies a stable voltage of 3.7V to the DUT and samples the power consumption at a rate of 5KHz.

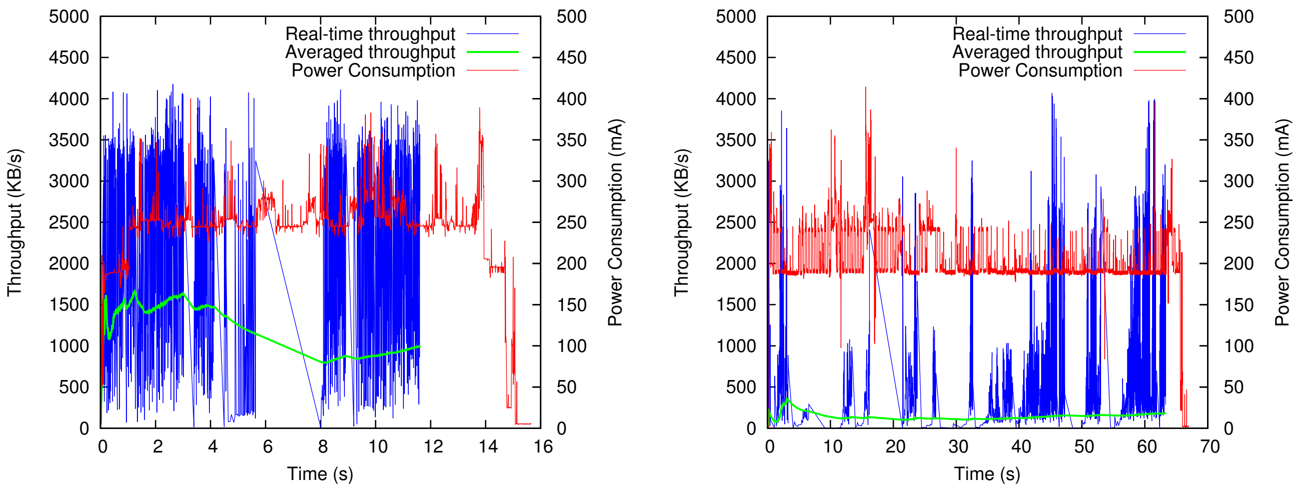
To perform the download, an application (WiDownload) that downloads files using the DownloadManager API provided by Android is developped and installed in the DUT. We also cross compile Iptables and Tcpdump and install them into the DUT as local libararies. We use Iptables to block the Internet access of all applications except that of WiDownload so that no interference traffic is introduced. During the download, the screen is off and the WifiLock is aquired. We measure the power consumed for completing downloading a file of 11.4 MB. Each measurement is repeated 3 times. After that, we move the instruments to another location where Rssi is different and reconduct the experiment.

B. Measurement Results

Figure 1 shows the power consumption for download a file at strong (-30 dBm) and weak (-84 dBm) Rssi and the corresponding traffic throughputs. Comparing the two, we can infer that the energy consumption for downloading files can be assumed to be proportional to the time it takes to finish the download. Rssi influences the the energy consumption for downloading files mainly due to its impact on the throughput. The whole dataset of the relationship between Rssi (averaged over time) and the energy consumption is plotted in Figure 2.

We conclude that when Rssi is higher than -70dBm, the energy consumption for the download is limited and independent of Rssi. When Rssi is between -80dBm and -70dBm, the energy consumption for the download increases as Rssi becomes weaker, but not so significantly. When Rssi is below -80dBm, the energy consumption for downloading files increases dramatically.

Comparing the duration of throughput and energy consumption, we can see that the energy consumption duration is always 3-5 seconds longer than that of throughput. To further investigate this phenomenon, we use WiDownload to download a 22KB file, which is downloaded in less than 3 seconds (establishing of TCP connection included), but the duration of energy consumption is more than 6 seconds, as shown in Figure 3. We get similar results when downloading files in the interval of 10KB and 700KB. We conclude that



(a) Strong Rssi (-30dBm)

(b) Weak Rssi (-84dBm)

Fig. 1: Comparison of power consumption and throughput for downloading a file of 11.4MB in strong and weak Rssi

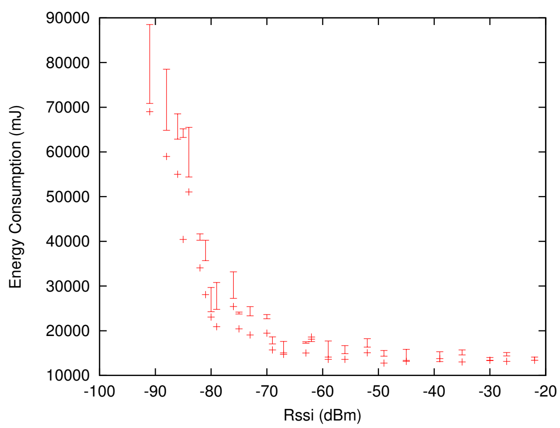


Fig. 2: Energy Consumption in different Rssi

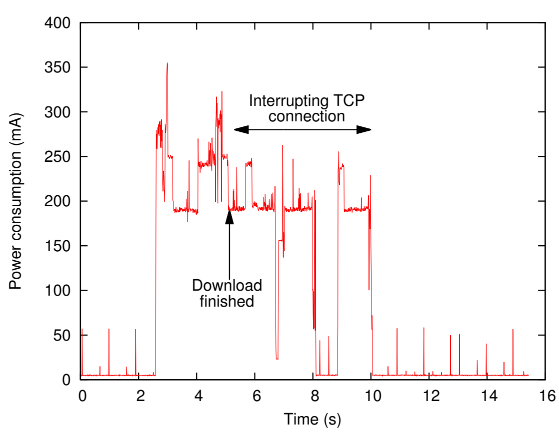


Fig. 3: Energy Consumption of downloading a 22KB file

downloading files via Wi-Fi introduces significant tail energy, which is due to the interruption of TCP connection. We can see that tail energy represents a great overhead if the download session is short.

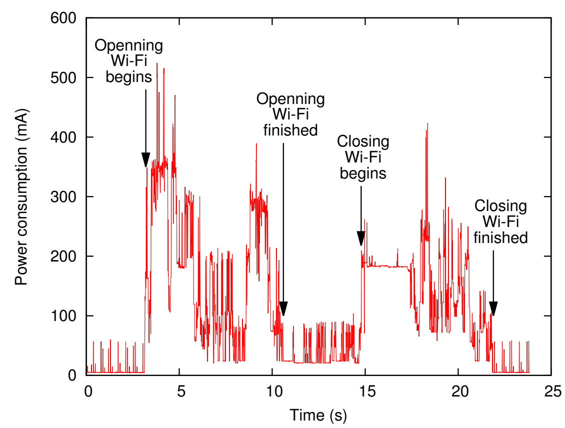


Fig. 4: Energy Consumption of openning and closing Wi-Fi interface

III. Rssi-based Wi-Fi Download Management Algorithm

As Rssi has an significant impact on the energy consumption of Wi-Fi, one may consider closing the Wi-Fi interface when Wi-Fi signal is weak and reopening it when Wi-Fi signal is strong. However, our experiments show that closing and openning the Wi-Fi interface consumes much energy, 3700 mJ and 4300 mJ respectively, shown in Figure 5. Keeping the Wi-Fi interface open, on the other hand, consumes only 3.95 mW overhead. The energy consumption of closing and openning the Wi-Fi interface is equal to that of keeping the Wi-Fi interface open for 2025 seconds. Besides, keeping Wi-Fi open allows to monitor Wi-Fi condition and adjust the smartphone's networking behaviors accordingly. In designing our Rssi-based download management algorithm, we keep the Wi-Fi interface open.

We divide Wi-Fi signal strength measured by the smartphone into 3 categories: strong, relatively weak and weak, according to its impact on energy consumption. We define 3

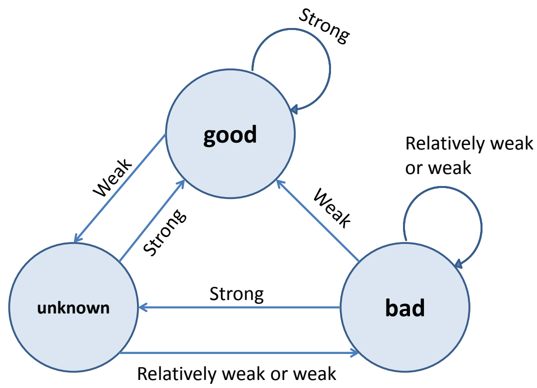


Fig. 5: FSM for updating current state of Wi-Fi

different states of Wi-Fi: *good*, *unknown* and *bad*. However, what's subtle is that the states of Wi-Fi are not a simple mapping of the signal strength of Wi-Fi, but obtained dynamically according to previous state and the Rssi measured from a FSM, as shown in Figure 5.

According to the current state of Wi-Fi, we adjust the download strategy adaptively. The basic idea is to start downloading files when the state of Wi-Fi is *good*, and to stop the download when the state is *bad*. If the state is *unknown*, we continue the download if the download is running, and stay in the paused state if the download is paused. The pseudo-code of our Rssi-based download management algorithm is given in Algorithm 1.

Algorithm 1 Rssi-based Download Management Algorithm

```
while true do
    /*Update current state of Wi-Fi,
    *as shown in Figure 5 */
    curr_state ← update_state(prev_state, rssi_measured)
    if curr_state is good then
        start_download()
    else if curr_state is bad then
        stop_download()
    end if
    prev_state ← curr_state
    sleep() //sleep for 1 second
end while
```

The main purpose of our FSM-based algorithm is to leverage the varying nature of Rssi. As each measurement could introduce inaccuracy, the Wi-Fi state *unknown* is set so as to leave the decision-making in the next iteration when we are not certain whether the change of Rssi is due to the smartphone user moving or due to the fluctuations of Wi-Fi signal itself. In this way, we avoid constantly starting and stopping downloads. This is effective because the relationship between Rssi and energy consumption is threshold-based, and thus the delay of 1 second in decision-making is tolerable.

Our algorithm is light-weighted and causes negligible energy overhead. In the one hand, we simply need to store Wi-Fi state of the previous iteration and each decision-making takes only 1 comparison. In the other hand, the Rssi calculation is done by the Android OS and WifiService and can be get through API provided by WifiManager. In fact, once Wi-Fi is connected to an AP, WifiService will be started in SystemServer and Rssi will be calculated periodically. We implemente our algorithm as an application. We measure the power consumption of running the application and not running the application, and the difference cannot be measured because it's too insignificant, far less than 0.001mA. Therefore, we can consider the energy overhead of our algorithm as zero.

IV. Simulation Results

A. Simulation Methodology

Halton Sequence: To simulate all possible starting point that a smartphone user could take, a 2-dimensional Halton sequence [6] with base {2, 3} is generated. When the user get out of the signal range of the Wi-Fi AP, we consider one walk is terminated and take the next value of the Halton sequence as his new starting point. This process is characterized as:

$$\vec{P}_{k_{t=0}} = R\mathrm{H}_k(2,3) \tag{1}$$

where R refers to the rayon of the signal range of the Wi-Fi AP.

Stationary-Moving State Transition: Considering that a smartphone user may use his phone not only in the move, but also when he is stationary, we introduce a Markov chain to characterize this stationary-moving transition, with two state spaces *moving* and *stationary* and the transition matrix:

$$A = \begin{pmatrix} p_m & 1-p_m \\ 1-p_s & p_s \end{pmatrix} \tag{2}$$

where p_m refers to the probability of the user continuing moving the next instant if he is on the move the current instant, and p_s refers to the probability of the user staying stationary the next instant if he is stationary the current instant. We represent the *moving* state as 1 and the *stationary* state as 0, noted by $\delta \in \{0, 1\}$.

Directional Random Walk: To simulate the process of walk of a smartphone user, we introduce the directional random walk defined as:

$$s_{k_t} \sim \mathrm{U}(s_{min}, s_{max}) \tag{3}$$

$$\theta_{k_t} \sim \mathrm{N}(\theta_{k_{t-1}}, \sigma^2) \tag{4}$$

where s_{k_t} and θ_{k_t} are respectively the speed and the heading of the smartphone user at instant t. Thus the position of phone at the next instant can be given as:

$$\vec{P}_{k_{t+1}} = \vec{P}_{k_t} + \delta_{k_{t+1}} s_{k_t} \begin{pmatrix} \cos\theta_{k_t} \\ \sin\theta_{k_t} \end{pmatrix} \tag{5}$$

Rssi Variation Model: We adopt the logarithmic wireless path model to calculate the real Rssi of Wi-Fi from the distance between the AP and the smartphone:

$$Rssi(d) = Rssi(d0) + 10\eta \log(d/d0) + \xi \tag{6}$$

However, as each measurement of Rssi could introduce inaccuracy, we define Equation (7)-(9) to describe the variations of Rssi in measurements.

$$v_{k_t} \sim \mathrm{N}(0, \phi^2) \tag{7}$$

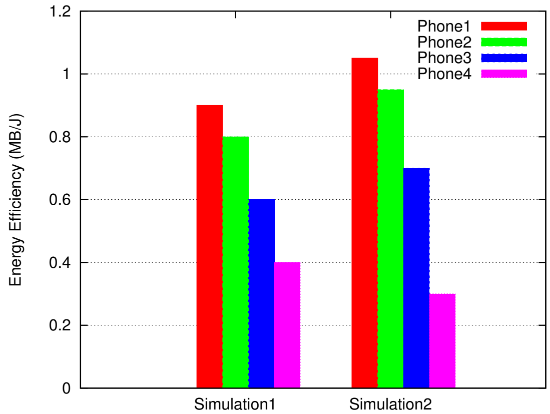


Fig. 6: Energy efficiency of the 4 phones in Simulation1 and Simulation2

$$\Delta_{k_t} \equiv \lfloor \|v_{k_t}\| \rfloor \pmod M \tag{8}$$

$$Rssi_{measured\_k_t}(d) = Rssi_{k_t}(d) + \text{sign}(v_{k_t})\Delta_{k_t} \tag{9}$$

where M refers to the maximum variation of measured Rssi from the real Rssi.

Monte-Carlo Simulation: We simulate a person carrying 4 phones with him and performs 1,000,000 directional random walks in a plan with an AP in the center. Phone1 runs our algorithm for the download management. Phone2 starts the download if Rssi is measured to be strong and stops the download if Rssi is measured to be weak or relatively weak. Phone3 starts the download if Rssi is measured to be strong or relatively weak and stops the download if Rssi is measured to be weak. Phone4 has no download strategy and keeps downloading regardless of Rssi. The 4 phones are all powered with a infinite-capacity battery, but their power consumptions are recorded. They are all the four charged with the task of downloading a file of infinite size. We incorporate our measurement results in our simulations, including the relationship between energy consumption and Rssi, the relationship between throughput and Rssi, the tail energy and the variation of Rssi.

Two simulations are carried out. In the first simulation, the person walks always and thus $p_m = 1$, $p_s = 0$. In the second simulation, the person walks and stops, with $p_m = 0.8$, $p_s = 0.9$.

B. Simulation Results

Our first simulation shows that in the constantly moving scenario, Phone1 will have 89.1% of its download time spent in strong Wi-Fi signal area, 10.1% in areas relatively weak Wi-Fi signal area. In the stationary-moving scenario, these two rates are 96.6% and 3.3%. In both simulations, Phone1 has 0% of its download time spent in weak Wi-Fi signal area. The rate of being in strong Wi-Fi signal area and not downloading is 0.13% in Simulation1 and 0.06% in Simulation2. We define the energy efficiency as the fraction of the size of file downloaded and the energy consumed, given in Equation (10):

$$EnergyEfficiency = \frac{SizeOfFileDownloaded}{EnergyConsumed} \tag{10}$$

The energy efficiency of the 4 phones are illustrated in Figure 6.

C. Analysis of Simulation Results

The performances of Phone2, Phone3 and Phone4 descrease, showing that even simple download controls according to Rssi can be effective. We further analysis why Phone1 outperforms Phone2.

In Simulation1, Phone2 has 13.5% more sessions than Phone1. This rate increases to 18.8% in Simulation2. More session number means more frequent transition between starting and stopping downloads, and wasting energy consumption in interrupting (and reestablishing) TCP connections, which causes considerable tail energy. At the same time, Phone1 also has 14.7% longer average download session length than Phone2 in Simulation1 and 21.4% in Simulation2. All these contribute to the better performance of Phone1 in preference to Phone2. We conclude that our algorithm is effective in energy saving and succeed in leveraging the variations of Rssi.

V. Conclusion

In this paper, we first investigate the relationship between Rssi and energy consumption of smartphones and then propose a novel Rssi-based Wi-Fi download management algorithm for energy savings. Our algorithm, as simple as it is, incorporates some major characteristics of Wi-Fi signal. Through Monte-Carlo simulations, we prove the effectiveness of our algorithm and conclude that it leverages some main challenges such as varying nature of Rssi and tail energy when stopping downloading.

Acknowledgment

This work is supported by the National Nature Science Foundation of China, NSFC (Grant No. 61170293) and the International Science & Technology Cooperation Program of China, Ministry of Science and Technology of China (Grant No. 2014DFG12370).

References

[1] Frank Vanheel et al., *Automated linear regression tools improve RSSI WSN localization in multipath indoor environment*, Journal on Wireless Communications and Networking, 2011

[2] Noah Pritt, *Indoor Positioning with Maximum Likelihood Classification of Wi-Fi Signals*, SENSORS 2013 IEEE, November 2013, p1-p4

[3] Doug Rosener, *Decreasing Rssi Setting Time in Low-power Mode Systems*, US Patent, Pub.No: US 2013/0029607 A1, Pub.Date: Jan 31, 2013

[4] Jenq-Shiou Leu, Nguyen Hai Tung, and Chun-Yao Liu, *Non-Parametric RSS Prediction Based Energy Saving Scheme for Moving Smartphones*, IEEE Transactions on Computers, July 2014, p1793-p1801

[5] Monsoon Power Monitor, *www.msoon.com/LabEquipment/PowerMonitor*

[6] Halton Sequence, *en.wikipedia.org/wiki/Halton_sequence*