An Rssi-Aware 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 an Rssi-aware Wi-Fi download management approach for energy saving on smartphones. Simulations show 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. We consider the scenario where people use their smartphones to download large files such as movies, documents and apk files. Those files are typically larger than 10 MB and may take several minutes to complete the download. The key challenge here is how to perform the download activity in an energy-efficient way so that it will not affect other applications.

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 scheme that can perceive the varying Rssi and automatically adjust the download actions. We conduct evaluation of our scheme using simulations.

II. Evaluation of Rssi and Energy Consumption

This section describes the experimental setup for power measurement and the analysis of measurement results.

*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 29, 2014.

A. Experimental Setup for Power Measurement

The DUT (Device Under Test) is a Huawei 8950D running Android 4.0, equiped with a double-core 1.2GHz Snapdragon CPU and 768MB RAM, supporting IEEE 802.11 n/b/g. We use a Monsoon Power Monitor [5] 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, we developed an application using the DownloadManager API provided by Android. 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 our application performing the download 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, which is the typical size of an apk file. Each measurement is repeated 3 times. After that, we move the instruments to another location where Rssi is different and repeat the experiment.

B. Measurement Results

Figure 1 shows the relationship between Rssi and the energy consumption. We can see 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. There is a clear threshold-based relationship between the Rssi and the energy consumption.

Figure 2 and Figure 3 show the measured power data for downloading a file at strong (-30 dBm) and weak (-84 dBm) Rssi and the corresponding traffic throughputs. The measured power data is mainly in the range of [250mA, 300mA] at strong Rssi and in the range of [200mA, 250mA] at weak Rssi. While the measured power data is slightly higher at strong Rssi than at weak Rssi, the throughput at strong Rssi is more than 5 times higher than at weak Rssi (1006KB/s vs 183KB/s), resulting in the signficant difference of time needed (11.4s vs 64s) and energy consumption (13312mJ vs 54397mJ) for completing the download . 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.

Comparing the duration of throughput and energy consumption, we can see that the energy consumption duration is

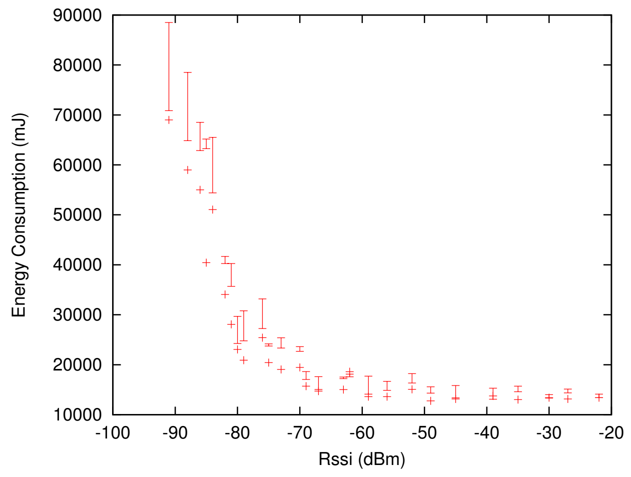

Fig. 1: Energy Consumption at different Rssi

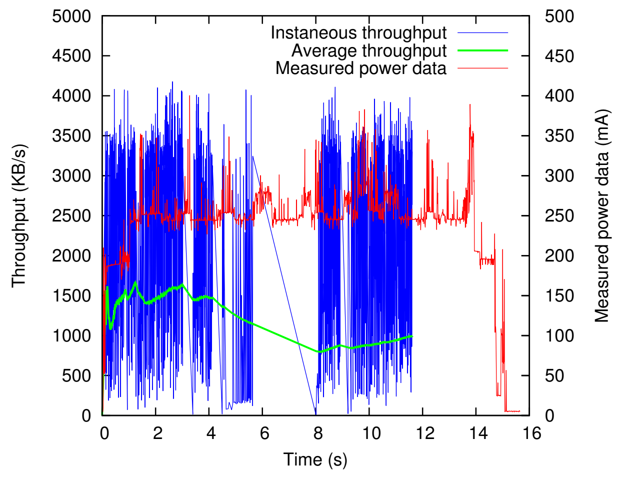

Fig. 2: Energy consumption vs throughput and Rssi (-30dBm)

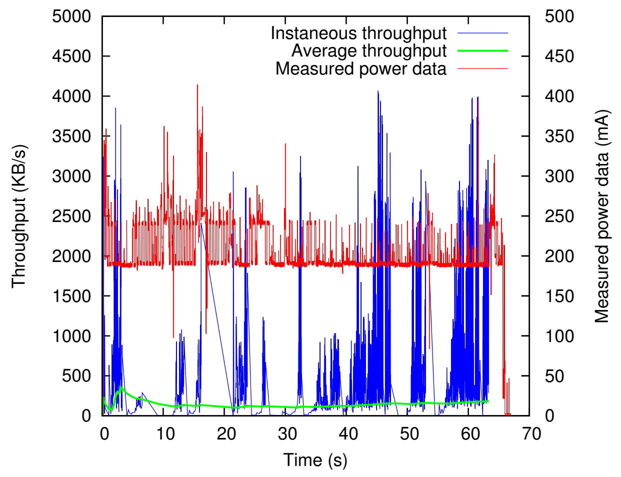

Fig. 3: Energy consumption vs throughput and Rssi (-84dBm)

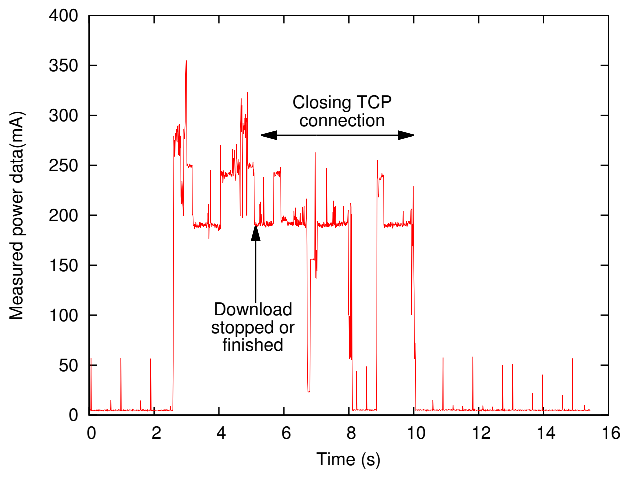

Fig. 4: Tail energy due to the closure of TCP connection when stopping or finishing a download

always 3-5 seconds longer than that of throughput. We have the same observation when downloading small files (in the order of 10KB and of 100KB) and when stopping downloading files. Our analysis using Tcpdump shows that the extra 3-5 seconds of energy consumption is caused by the closure of TCP connection, as illustrated in Figure 4. We conclude that downloading files via Wi-Fi introduces significant tail energy when stopping or finishing a download. We can see that tail energy represents a great overhead if the download session is short. Since this influences the energy consumption of downloading files significantly, we will consider this factor in our approach.

III. Rssi-based Wi-Fi Download Management Algorithm

As Rssi has a significant impact on the energy consumption of file downloading in Wi-Fi environment, one may consider closing the Wi-Fi interface when Wi-Fi signal is becoming weak and reopening it when Wi-Fi signal is becoming strong. However, our experiments show that closing and openning the Wi-Fi interface consumes much energy, 3700 mJ and 4300 mJ respectively, as shown in Figure 5. Keeping the Wi-Fi interface open, on the other hand, consumes only 3.95 mW overhead. As the result, 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. To this end, in designing our Rssi-based download management algorithm, we keep the Wi-Fi interface open.

According to the experimental results, we divide Wi-Fi signal strength into 3 status: strong, relatively weak and weak, based on its impact on energy consumption. Accordingly, we define 3 different states of Wi-Fi: *good*, *unknown* and *bad*. The transition among the three states is illustrated as a Finite State Machine (FSM) in Figure 6. At each iteration, we update the current state of Wi-Fi according to the FSM.

We adjust the download behavior according to the current state of Wi-Fi. The basic idea is to start downloading files

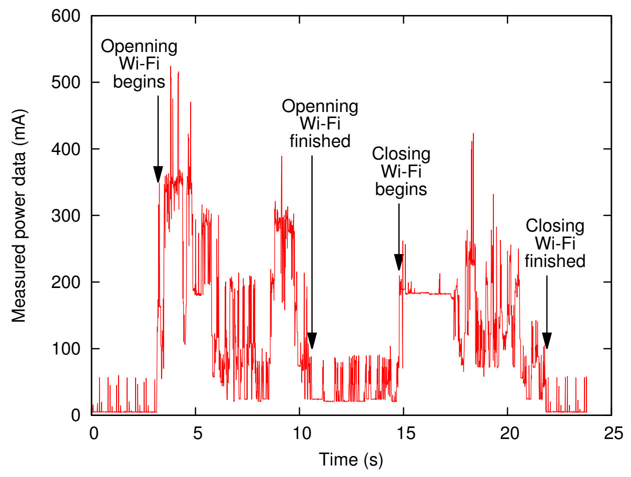

Fig. 5: Energy Consumption of openning and closing Wi-Fi interface

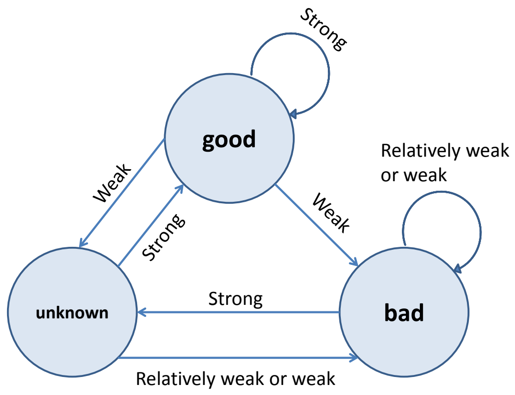

Fig. 6: FSM for updating current state of Wi-Fi

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

```
1:  while true do
2:      /*Update current state of Wi-Fi,
3:      *as shown in Figure 6 */
4:      curr_state ← update_state(prev_state, rssi_measured)
5:      if curr_state is good then
6:          start_download()
7:      else if curr_state is bad then
8:          stop_download()
9:      end if
10:     prev_state ← curr_state
11:     sleep() //sleep for 1 second
12: end while
```

The main purpose of our 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. On the one hand, we simply need to store Wi-Fi state of the previous iteration and each decision-making takes only 1 comparison. On the other hand, the Rssi calculation is done by the Android OS and WifiService and can be obtained 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. The energy overhead of running the application 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

In this section, we present our Monte Carlo simulation of our Rssi-aware Wi-Fi download management approach and the simulation results.

A. Simulation Methodology

User's Moving Behavior Model: We model the moving behavior of a smartphone user as a directional random walk. To simulate all possible starting point that the 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.

Considering that the 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. We denote p_m as the probability of the user continuing moving the next instant if he is on the move the current instant and p_s as the probability of the user staying stationary the next instant if he is stationary the current instant.

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

$$s_{k_t} \sim \mathrm{U}(s_{min}, s_{max}) \tag{1}$$

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

where s_{k_t} and θ_{k_t} are respectively the speed and the heading of the smartphone user at instant t. Representing the moving state as 1 and the stationary state as 0, noted by $\delta \in \{0, 1\}$, the position of phone at the next instant can be given as:

$$\vec{P}_{k_{t+1}} = \vec{P}_{k_t} + \delta_{k_{t+1}} s_{k_t} \begin{pmatrix} \cos\theta_{k_t} \\ \sin\theta_{k_t} \end{pmatrix} \tag{3}$$

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:

$$Rssi(d) = Rssi(d0) + 10\eta \log(d/d0) + \xi \tag{4}$$

where d0 is the reference distance, typically 1 meter, η is the path attenuation factor and ξ the shadowing factor. However, as each measurement of Rssi could introduce inaccuracy, we define Equation (5)-(7) to describe the variations of Rssi in measurements.

$$v_{k_t} \sim \mathrm{N}(0, \phi^2) \tag{5}$$

$$\Delta_{k_t} \equiv \lfloor \| v_{k_t} \| \rfloor \pmod{M} \tag{6}$$

$$Rssi_{measured\_k_t}(d) = Rssi_{k_t}(d) + \mathrm{sign}(v_{k_t})\Delta_{k_t} \tag{7}$$

where M refers to the maximum variation of measured Rssi from the real Rssi and is given as:

$$M = \lfloor \| \frac{Rssi_{k_t}(d)}{10} \| \rfloor - 1 \tag{8}$$

Monte-Carlo Simulation: We simulate a smartphone user 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 energy consumptions are recorded. We assume that the file that the 4 phones download is of infinite size.

Two simulations are carried out. In the first simulation, the user walks always and thus p_m = 1, p_s = 0. In the second simulation, the user 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 as:

$$EnergyEfficiency = \frac{SizeOfFileDownloaded}{EnergyConsumed} \tag{9}$$

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

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 closing (and reestablishing) TCP connections, which causes

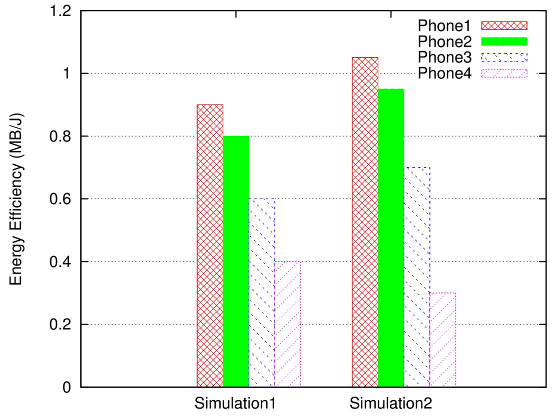

Fig. 7: Energy efficiency of the 4 phones in Simulation1 and Simulation2

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 an Rssi-aware 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*