An Rssi-Based Wifi Access Management Approach for Energy Saving on Smartphones

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

***Abstract*—The abstract goes here.**

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

I. Introduction

GUIDANCE: This paragraph is to address 1) the growth of the multimedia applications on smartphone [Need some data evidence], and most need download large files, e.g., app tar files and movies etc. 2) the wifi access requirements for those applications, and 3) major Wifi properties and the impact on the critical energy issues on resource-constrained smartphones.

[**Guidance:]** 1) The most widely used energy-efficient wifi access strategies, such as prediction sleeping methods are introduced here, 2) The limits or problems of those strategies.

Energy optimization for smartphones has attracted considerable attention in recent years, and lots of researches have been done on the energy consumption of Wi-Fi.

Tan et al. [1] proposed power save mode (PSM) throttling, an application-independent protocol, which reshapes TCP traffic into periodic bursts with the same average throughput as the server transmission rate. Clients accurately predict the arriving time of packets, and turn on/off the wireless interfaces accordingly. PSM-throttling can minimize power consumption on TCP-based bulk traffic by effectively utilizing available Internet bandwidth without degrading the performance of application perceived by the user.

Fahad R. Dogar et al. [2] proposed Catnap, a system that exploits high bandwidth wireless interfaces by combining small gaps between packets into meaningful sleep intervals, thereby allowing the NIC as well as the device to doze off. Catnap targets data oriented applications, such as web and file transfers, which can afford delay of individual packets as long as the overall transfer times do not increase. Catnap is implemented as an application independent proxy to decouple wired and wireless segments, and schedules ADU transmissions on the wireless side using hints regarding the application workload.

In [3], Young-Seol Lee and Sung-Bae Cho proposed a context-aware battery management system that uses probabilistic models to infer a user's situation with low power sensors such as acceleration and orientation and controls components that consumes a large amount of energy. For example, when the system infers that the user goes out, it turns off Wi-Fi automatically. However, the authors didn't take into consideration the energy overhead of closing and reopening the Wi-Fi interface, which is more than 30 minutes of the energy consumption of keeping it open.

In [4], Jian Li et al. investigate the key factors influencing Wi-Fi energy consumption, and propose three energy management schemes: 1) Dynamic control of Wi-Fi on/off interface; 2) improve communication efficiency via application packing; and 3) elongation of Wi-Fi Power Save Mode (PSM) via application alignment under mixed application workload. They design and test their solution as a device-side application utilizing general system process scheduling and network firewall techniques. However, their work regarding to Rssi didn't take into consideration the energy consumption characteristics, geometric characteristics and variation characteristics of Wi-Fi.

Previous work that is the most similar to ours is [5], which proposed a non-parametric Rssi prediction based energy-saving scheme for moving smartphones. By periodically monitoring the Rssi in diverse network environments, their proposed scheme applies weighted scatter plot smoothing and kernel moving average algorithms to adaptively adjust file downloading and video streaming rates. However, the KMA prediction model that they adopted consumes much energy, causing a significant energy overhead. Besides, in the interval of [-90dBm, -70dBm], the variation of Rssi is great, and the accuracy of prediction models such as KMA can be greatly degraded. As they showed, in a fixed location scenario, their scheme may consume more than 2 times energy than if no prediction or adaption adopted for downloading files.

[**Guidance:]** The contributions of this paper: 1) extensive experiments to evaluate the relationship of rssi and the energy consumption, 2) propose a novel Rssi-based wifi access management algorithm, 3) effectiveness evaluation using simulations.

II. Evaluation of Rssi and Energy Consumption

A. Experimental Setup for Power Measurement

This section describes the experimental setup for power measurement.

Experimental Setup and Measurement Technology: 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 MSM8225 CPU and 768MB RAM. We use a Monsoon Power Monitor 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.

*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.

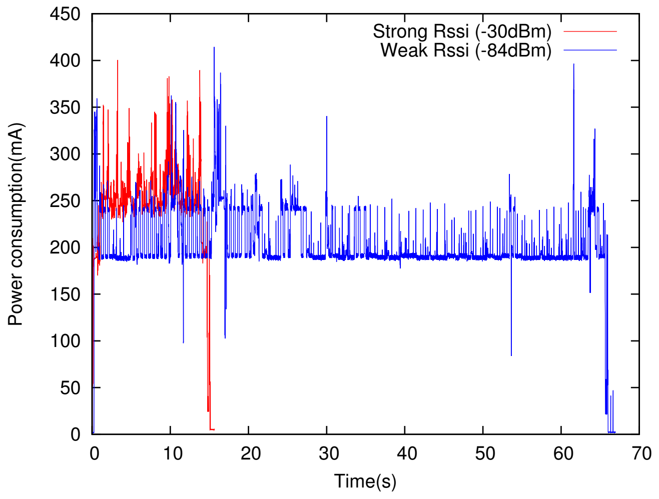

Fig. 1. Power consumption for downloading a file of 11.4Mb in different Rssi

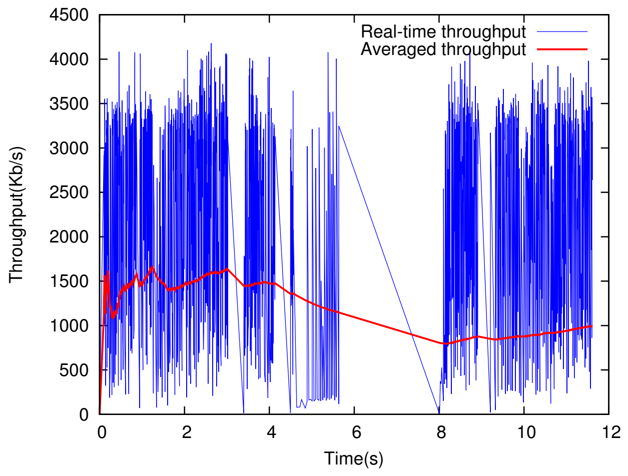

Fig. 2. Throughput in strong Rssi (-30dBm)

Measurement Methodology: 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. Iptables is used to block the Internet access of unrelated applications to avoid interference traffic. Tcpdump is used to analyze traffic performance such as RTT, packet loss and throughput.

We adjust the distance between the DUT and the AP to get different Rssi. When we perform measurements, we keep all instruments at fixed location. 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. After this, we perform the same download at the same fixed location, but with Tcpdump running to dump traffic information. Each measurement is repeated 3 times. After that, we move the instruments to another location where the 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. Figure 2 and 3 are 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 4.

We conclude that when Rssi is higher than -70dBm, the energy consumption for the download is limited and independent of Rssi. When the Rssi is between -70dBm and -80dBm, the energy consumption for the download increases as Rssi

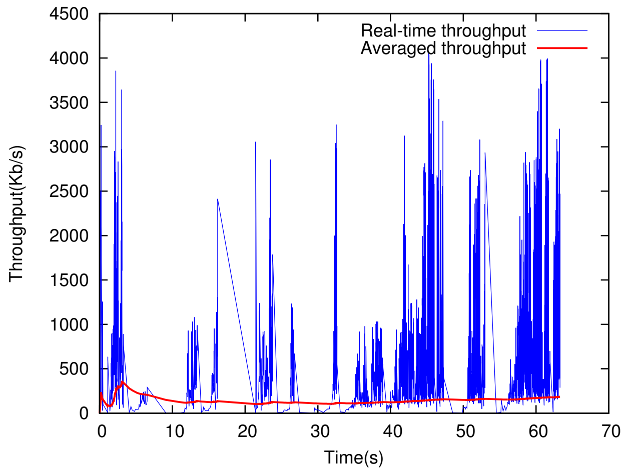

Fig. 3. Throughput in weak Rssi (-84dBm)

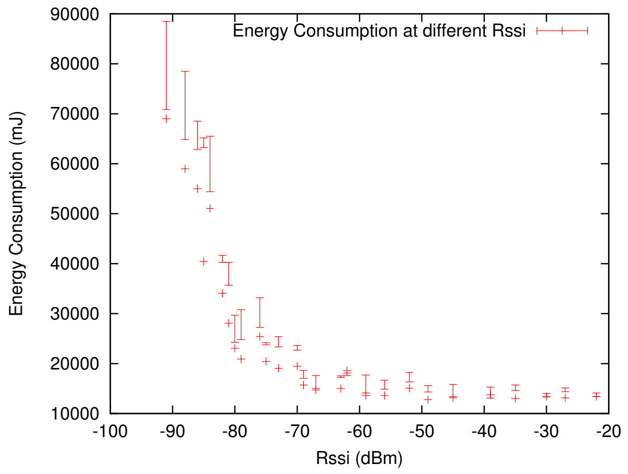

Fig. 4. Energy Consumption in different Rssi

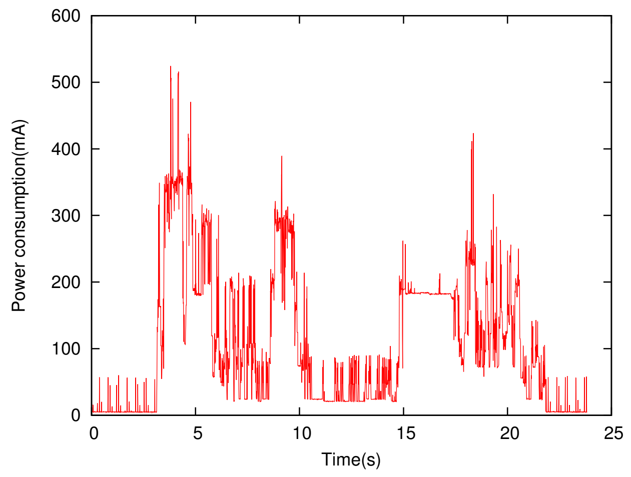

Fig. 5. Energy Consumption of openning and closing Wi-Fi interface

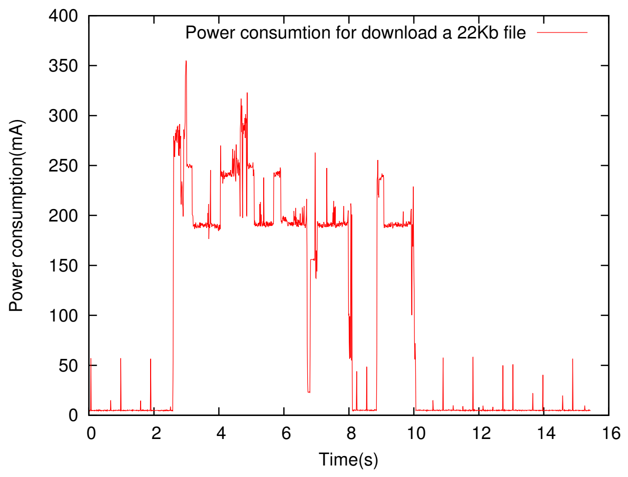

Fig. 6. Energy Consumption of downloading a 22KB file

becomes weaker, but not so significantly. When the 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 in 1 second, but the duration of energy consumption is more than 6 seconds. We conclude that Wi-Fi, as with 3G, introduces significant tail energy. We can see that tail energy represents a great overhead if the download session if short.

III. Rssi-based Wi-Fi Download Control Algorithm

As Rssi has an significant impact on the energy consumption of the Wi-Fi downloading, one may consider closing the Wi-Fi interface when the Wi-Fi signal is weak and reopening it when the 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 open for 2025 seconds. Besides, keeping the Wi-Fi open allows to monitor Wi-Fi condition and adjust the smartphone's networking behaviors accordingly. In designing our algorithm, we keep the Wi-Fi interface open.

Based on our finding in experiments, we are able to conceive the Rssi-based Wi-Fi Download Control Algorithm. 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:

(A) Strong: Rssi $>=$ -70dBm

(B) Relatively weak: Rssi $\in$ [-80dBm, -70dBm)

(C) Weak: Rssi $<$ -80dBm

We define 3 different perceptions of Wi-Fi: *good*, *acceptable*, *bad*. However, what's subtle is that the perception of Wi-Fi is not a simple mapping of the signal strength of Wi-Fi, but obtained dynamically according to previous perception and the Rssi measured, shown in Figure 5 and given in Algorithm 1.

According to the current perception of Wi-Fi, we adjust the download strategy adaptively, shown if Figure 6 and given in Algorithm 2. The basic idea is to start downloading files when the perception of Wi-Fi is *good*, and to stop the download when the perception is *bad*. If the perception is *acceptable*, we continue the download if the download is running, and stay in the paused stated if the download is paused.

Our Rssi-based Wi-Fi Download Control Algorithm works iteratively in three steps: update current perception of Wi-Fi, ajust download according to the current perception of Wi-Fi and sleep for 2 seconds. The pseudo-code in given in Algorithm 3.

Our algorithm is light-weighted. We simply need to store the perception of Wi-Fi of the previous iteration. The decision-making takes only 2 comparisons. It takes into consideration three major characteristics of Rssi of Wi-Fi.

First, when Rssi is higher than -70dBm, the energy consumption for downloading files varies little and can be considered as constant. When Rssi is between -70dBm and -80dBm, the energy consumption for downloading files increases but not so significantly. So we can tolerate some time of download spent in the range of -70dBm and -80dBm if this could lead to better overall energy-saving strategy.

Second, we should also take into consideration the geometric characteristics of the Wi-Fi signal. Our measurements show that the distance of the Rssi range of -70dBm to -80dBm is 4-10 meters, depending on the obstacles present in the Wi-Fi environment. As a person walks typically around 1.5m/s, not necessarily against the rayon direction of the Wi-Fi signal of AP, the phone has always sufficient time to adjust its download behavior to the Wi-Fi signal change within one sampling period, 2 seconds in our algorithm.

Third, when the Wi-Fi signal is weak or relatively weak, the variation of the Rssi measured is significant. Each measurement of Rssi can thus be inaccurate. Adapting the download behavior to the Rssi change should leave space for the verification of the Rssi.

We incorporate these ideas in our algorithm mainly in the decision tree for updating current perception of Wi-Fi. The path 1, 3, 6 and 9 is easy to understand. Here are further analysis of the other 5 path:

(a) Path 2: The previous perception is *good*, according to Algorithm 2, the phone is downloading files. As each Rssi measurement could be inaccurate, the Rssi measured is not sure indicator that the Wi-Fi signal is degraded. We set Path 2 to be *acceptable*, so that the decision-making is to be performed at the next iteration. If the relatively weak Rssi is due to accidental measurement error, the next moment Path 4 will be taken and the perception will be updated to be *good*. In this way, the download is not interrupted and the energy overhead of reestablishing TCP connection is spared. If the relatively weak Rssi is due to the user moving to where the signal is degraded, the path 5 or 6 will be taken the next moment and the download will be stopped. This idea coincides with the ZHUANLI.

(b) Path 4: As the previous perception is *acceptable*, the perception prior to the previous moment could be *good* or *bad*. If prevprevperception is *good*, the Path 2 is taken at the previous moment. So the sequence of the 3 most recent Rssi measured is: strong $\rightarrow$ relatively strong $\rightarrow$ strong. The download is not interrupted. If prevprevperception is *bad*, then the sequence of the 3 most recent Rssi measured is: relatively weak or weak $\rightarrow$ strong $\rightarrow$ strong. It's a good indicator the phone comes into the strong Wi-Fi zone from a relatively distant location where the Wi-Fi is weak or relatively weak. And the *good* set here will trigger the download to start.

(c) Path 5: As the previous perception prevperception is *acceptable*, the perception prior to the previous moment prevprevperception could be *good* or *bad*. If prevprevperception is *good*, the sequence of the 3 most recent Rssi measured is: strong $\rightarrow$ relatively weak $\rightarrow$ relatively Weak, which is a good indicator that we get far away from the AP. As Path 5 is set to be *bad*, the download will be stopped and waste of energy in a degraded Wi-Fi is avoided. If prevprevperception is *bad*, then the download is not started and the paused state of download will be continued.

(d) Path 7: The jump of the Rssi measured from weak to strong is rare, so this could be caused by the accidental measurement error. Not sure about the real state of the Wi-Fi, we leave the decision-making to the next moment, and either Path 4, Path 5 or Path 6 will be taken.

(e) Path 8: As the previous perception is *bad*, the download is not running. The download will not be started until two consequent strong Rssi is measured, which is good indicator that we get into the strong Wi-Fi signal area and two consequent measurements reduce the uncertainty of each one measurement.

Our algorithm causes negligible energy overhead. Because the Rssi calculation is done by the OS and WifiService.

IV. Simulation Results

In this section, we present our Monte Carlo simulation of the Rssi-based Wi-Fi Access Algorithm and its detailed analysis.

Algorithm 1 Update Rssi Perception

```
procedure UPDATEPERCEPTION
    switch prevPerception
        case good do
            switch measuredRssi
                case strong do
                    currPerception ← good
                end case
                case relativelyWeak do
                    currPerception ← acceptable
                end case
                case weak do
                    currPerception ← bad
                end case
            end switch
        end case
        case acceptable do
            switch measuredRssi
                case strong do
                    currPerception ← good
                end case
                case relativelyWeak do
                    currPerception ← bad
                end case
                case weak do
                    currPerception ← bad
                end case
            end switch
        end case
        case bad do
            switch measuredRssi
                case strong do
                    currPerception ← acceptable
                end case
                case relativelyWeak do
                    currPerception ← bad
                end case
                case weak do
                    currPerception ← bad
                end case
            end switch
        end case
    end switch
end procedure
```

Algorithm 2 Adjust Download According to Rssi Perception

```
procedure ADJUSTDOWNLOAD
    if notDownloading and currPerception is good then
        startDownload()
    else if isDownloading and currPerception is bad then
        stopDownload()
    end if
    prevPerception ← currPerception
end procedure
```

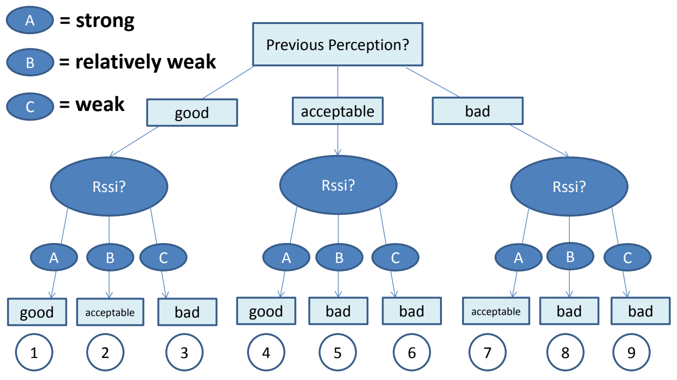

Fig. 7. Decision tree for updating current perception of Wi-Fi

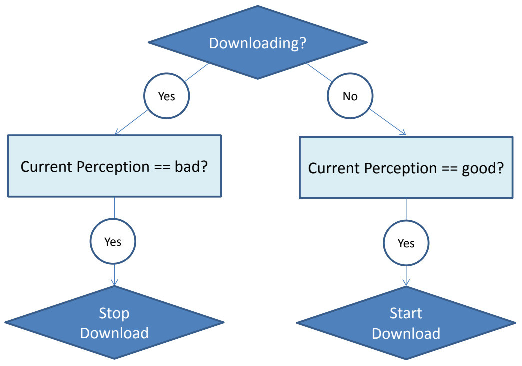

Fig. 8. Decison tree for dowload adjust according to current perception of Wi-Fi

A. Simulation Methodology

Halton Sequence: Halton sequences are sequences used to generate points in space for numerical methods such as Monte Carlo simulations. Although these sequences are deterministic they are of low discrepancy and appear to be random. In our simulation, to simulate all possible starting point that a smartphone user could take, a 2-dimensional Halton sequence 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. A starting point out of the range of the Wi-Fi is discarded and the user takes the $k+1^{th}$ walk.

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

$$s_{k_t} \sim \mathrm{U}(s_{min}, s_{max}) \tag{2}$$

$$\theta_{k_t} \sim \mathrm{N}(\theta_{k_{t-1}}, \sigma^2) \tag{3}$$

where s_{k_t} and θ_{k_t} are respectively the speed and the heading of the smartphone user at instant t, drawn respectively from an uniform distribution and a Gaussian distribution. $\theta_{k_{t=0}}$ is set to be 0 as the signal range of the AP is a circle and therefore symmetric. (...) Thus the position of phone at the next instant can be given as:

$$\vec{P}_{k_{t+1}} = \vec{P}_{k_t} + s_{k_t}\begin{pmatrix}\cos\theta_{k_t}\\ \sin\theta_{k_t}\end{pmatrix} \tag{4}$$

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:

$$A = \begin{pmatrix} p_m & 1-p_m \\ 1-p_s & p_s \end{pmatrix} \tag{5}$$

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. Representing the *moving* state as 1 and the *stationary* state as 0, noted by $\delta \in \{0, 1\}$, equation (4) can then be changed to:

$$\vec{P}_{k_{t+1}} = \vec{P}_{k_t} + \delta_{k_{t+1}} s_{k_t}\begin{pmatrix}\cos\theta_{k_t}\\ \sin\theta_{k_t}\end{pmatrix} \tag{6}$$

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:

$$Rssi(d) = Rssi(d0) + 10\eta\log(d/d0) + \xi \tag{7}$$

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 (8)-(10) to describe the variations of the Rssi in measurements.

$$d_{k_t} \sim \mathrm{N}(0, \phi^2) \tag{8}$$

$$\Delta_{k_t} \equiv \lfloor \| d_{k_t} \| \rfloor \pmod{M} \tag{9}$$

$$Rssi_{measured\_k_t}(d) = Rssi_{k_t}(d) + \mathrm{sign}(d_{k_t})\Delta_{k_t} \tag{10}$$

Where M refers to the maximum variation of measured Rssi from the real Rssi. For example, if the real Rssi is -20dBm and measured Rssi is {-19dBm, -20dBm, -21dBm}, then the M is 1. Our experiments show that M is 5 if $Rssi(d) \in$ (-70dBm, -60dBm], 6 if $Rssi(d) \in$ (-80dBm, -70dBm], 7 if $Rssi(d)$ is lower than -80dBm. So M is given as:

$$M = \lfloor \| \frac{Rssi_{k_t}(d)}{10} \| \rfloor - 1 \tag{11}$$

Monte-Carlo Simulation: We simulate a person carrying 4 phones with him and performs 1,000,000 directional random

Algorithm 3 Rssi-based Download Control Algorithm
```
1: procedure RssiBasedDownloadControl
2:     while True do
3:         UpdatePerception()
4:         AdjustDownload()
5:         Sleep() //sleep for 2 seconds
6:     end while
7: end procedure
```

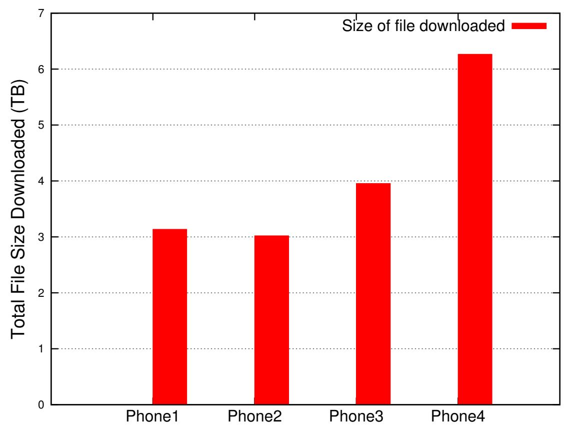

Fig. 9. Size of file downloaded in simulation 1

walks in a plan with an AP in the center. Phone1 runs our algorithm for the download control. Phone2 starts the download if the Rssi is measured to be strong and stops the download if the Rssi is measured to be weak or relatively weak. Phone3 starts the download if the Rssi is measured to be strong or relatively weak and stops the download if the Rssi is measured to be weak. Phone4 has no download strategy control and keeps downloading regardless of the 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

First Simulation: Our first simulation shows that in the constantly moving scenario, Phone1 will have 89.1% of its download time spent in areas where the Wi-Fi signal is strong, 10.1% in areas where the Wi-Fi signal is relatively weak, and 0% in areas where the Wi-Fi signal is weak. The rate of being in strong Wi-Fi area and not downloading is 0.13%.

The size of file downloaded and the energy consumed by the 4 phones are shown in Figure 9 and Figure 10. We define the energy efficiency as the fraction of the size of file downloaded and the energy consumed, given in Equation (12). The energy efficiency of the 4 phones in the constantly moving scenario is illustrated in Figure 11.

$$EnergyEfficiency = \frac{SizeOfFileDownloaded}{EnergyConsumed} \quad (12)$$

Second Simulation: Our second simulation shows that in the stationary-moving Markov chain scenario, Phone1 will have 96.7% of its download time spent in areas where the Wi-Fi signal is strong, 3.3% in areas where the Wi-Fi signal is relatively weak, and 0% in areas where the Wi-Fi signal is weak. The rate of being in strong Wi-Fi area and not downloading is 0.06%. The size of file downloaded, the energy consumed, and the energy efficiency of the 4 phones are respectively shown in Figure 12, Figure 13 and Figure 14.

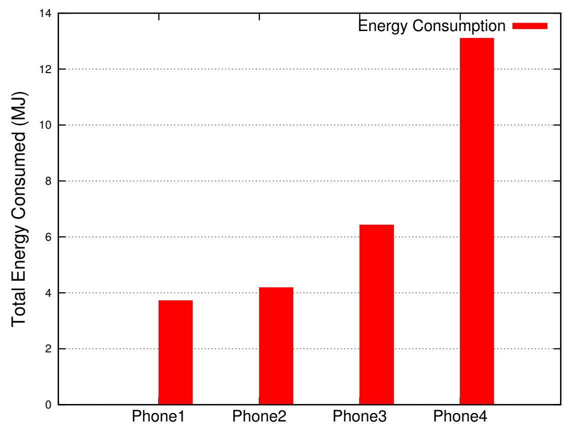

Fig. 10. Energy consumed for download in simulation 1

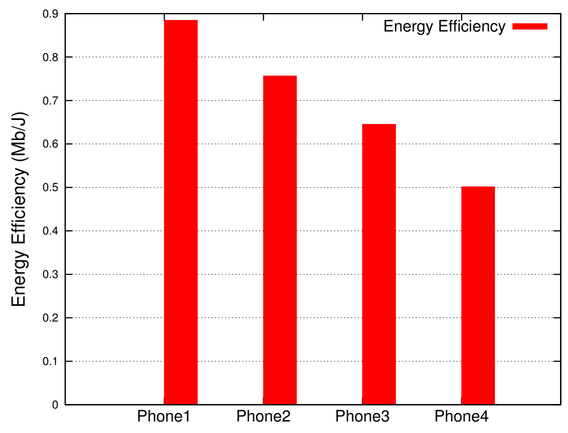

Fig. 11. Energy efficiency of the 4 phones in simulation 1

V. Conclusion

The conclusion goes here.

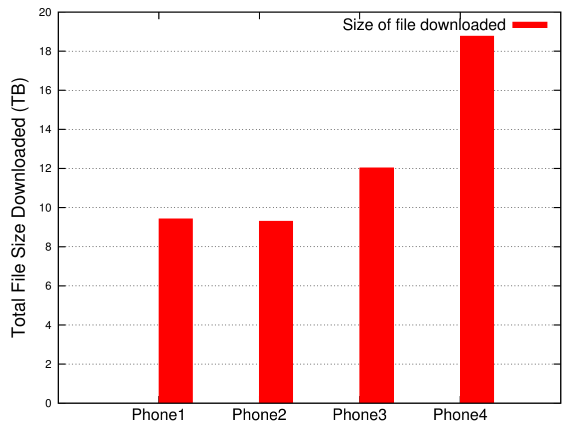

Fig. 12. Size of file downloaded in simulation 2

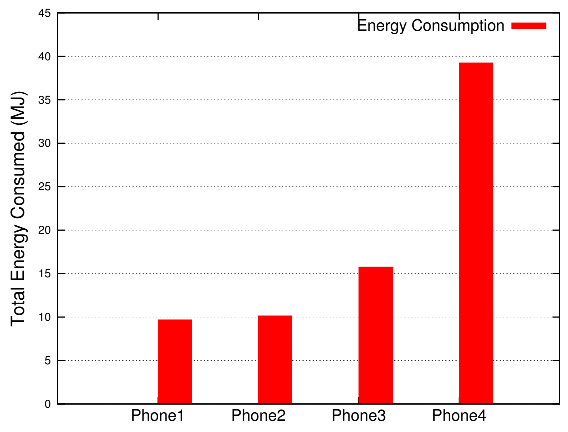

Fig. 13. Energy consumed for download in simulation 2

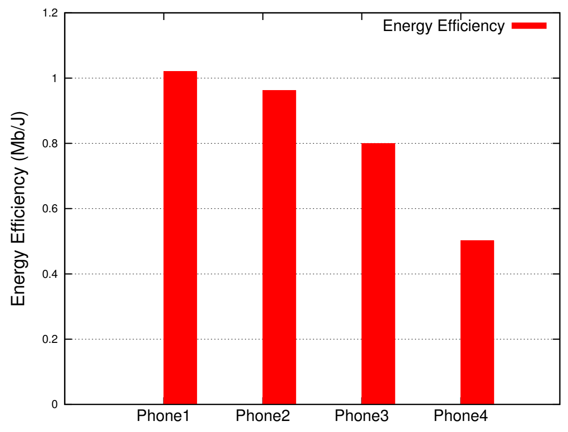

Fig. 14. Energy efficiency of the 4 phones in simulation 2

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] 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, VOL. 63, NO. 7, JULY 2014, p1793-p1801

[1] E. Tan, L. Guo, S. Chen, and X. Zhang, “Psm-throttling: Minimizing energy consumption for bulk data communications in WLANs,” in IEEE International Conference on Network Protocols, pp. 123–132, October 2007.

[2] Catnap: Exploiting High Bandwidth Wireless Interfaces to Save Energy for Mobile Devices

[3] An Efficient Energy Management System for Android Phone Using Bayesian Networks

[4] Application-Centric Wi-Fi Energy Management on Smart Phone

[5] Jenq-Shiou Leu, Nguyen Hai Tung, and Chun-Yao Liu, Non-Parametric RSS Prediction Based Energy Saving Scheme for Moving Smartphones