PowerGuide: Accurate Wi-Fi Power Estimator for Smartphones

Jian Li*, Jin Xiao†, Heni Azzouz§, James Won-Ki Hong*, Raouf Boutaba‡

*Division of IT Convergence Engineering, POSTECH, Korea
email: {gunine, jwkhong}@postech.ac.kr
†IBM T. J. Watson Research Center, US
email: jinoaix@us.ibm.com
‡David R. Cheriton School of Computer Science, University of Waterloo, Canada
email: rboutaba@cs.uwaterloo.ca
§Higher School of Communication of Tunis, Tunisia
email: azzouz.heni@supcom.rnu.tn

***Abstract*—Wi-Fi is a popular wireless communication technology for smart devices such as smartphones, however, Wi-Fi related energy consumption in smartphones contributes to a significant portion of its energy expenditure. Hence it is important to carefully analyze and profile Wi-Fi energy expenditure in order to improve mobile applications' energy efficiency. Although hardware based power meters provide very accurate power measurement result, it is infeasible to expect mobile application developers to rely on power meters. As alternative solutions, software based power estimation tools are popular. Most of the existing power estimation solutions to date do not provide accurate estimation result, as we disclose in this paper by analyzing the popular PowerTutor. Therefore, we propose a new Wi-Fi power model by taking into account important IEEE 802.11 communication patterns as well as Wi-Fi hardware settings in a way that increases the accuracy of power estimation. We design and implement the proposed power model as a smartphone application and deploy it into a real device for validation. The evaluation results show that our solution can achieve an estimation accuracy up to 86% compared to hardware power meters, and is much more accurate than PowerTutor.**

***Keywords*—Smartphones, Power Estimation, Wi-Fi**

I. Introduction

Wi-Fi is a popular wireless communication technology, used prevalently by smart devices such as laptops and smartphones. As networked smartphone applications proliferate and becoming a major operations leverage for Telecommunication operators (e.g., Wi-Fi hotspots can offload mobile traffic load) and smartphone users (e.g., almost free of charge), the market penetration ratio of Wi-Fi has been growing rapidly in the past decade, and such trend will continue in the coming years. Wi-Fi energy consumption is a significant portion of smartphone energy usage [1]. Therefore, smartphone operating system designers and Wi-Fi standardization groups have made lots of effort to provide a vast number of tunable options for application developers to design and implement their applications. However, in order to fully utilize these features, application developers need to use energy profiling tools to monitor the energy expenditure of their applications. Some research works such as [2][3] provide the solution for profiling the energy usages per smartphone application, but only restricted to CPU, display and did not consider Wi-Fi or other communication hardware. Furthermore, smartphone users may like to understand the energy consumption of their networking applications better not only in terms of bandwidth consumption but also energy consumption. In this context, this paper studies the problem of accurate Wi-Fi power monitoring for end users and application developers to understand the effect of their applications on the battery discharge behavior. Hardware based power meters provide accurate power measurement result, but they are expensive and impractical for regular uses.

Therefore, software based power estimators are highly popular. Existing software based power estimation tools (such as the popular PowerTutor [4], and research work in [5]) have low measurement accuracy due to the coarse grained power modeling and the lack of consideration for Wi-Fi hardware operation patterns. In general, Wi-Fi power estimation tools rely on a certain power model which is mainly determined by a set of power states (e.g., Continuous Active Mode: CAM and Power Saving Mode: PSM) under which the power consumption rates are very different. The power states are moderated by the Wi-Fi communication module based on the near term packet rate. By identifying the different power states of the Wi-Fi device, and measuring the total amount of time that the Wi-Fi device spend in each power state, it is feasible to calculate the overall Wi-Fi energy expenditure.

In this work, we found that 1) there are more than two power states implemented by contemporary Wi-Fi communication hardware; 2) the power state transition is not only triggered by the packet rate, but the packet inter-arrival time; and 3) the uplink and downlink power expenditure show asymmetric characteristics. These three findings are not properly considered in existing power estimation tools, and this in turn results low accuracy power estimation. Accordingly, we propose a new power model that considers important factors, and developed a new power estimation tool called PowerGuide. To derive the power model for Wi-Fi communication module, we first perform detailed experimental studies to characterize the power expenditure pattern, and then we detect the impact of the network conditions and parameters such as packet rate, packet size and packet inter-arrival time on the power con-

sumption profile. We then design a Finite State Machine (FSM) based power model based on our findings, and implement the power model as a mobile application for Android platform. Finally, we validate PowerGuide by deploying it on Android smartphone and conduct validation studies.

The remainder of this paper is structured as follows. Section II presents related works on Wi-Fi power estimation techniques, and with the help of hardware power monitoring tool - Monsoon, we study the characteristic of power consumption pattern of Wi-Fi by varying several network parameters in Section III. In Section IV, we show the detailed steps of deriving the Wi-Fi power model. Section V reports the evaluation result of the proposed Wi-Fi power model. Conclusion is given in Section VI.

II. Related work

Power modeling of smartphones in literature generally relies on one of the two methods: 1) constructing models based on battery behavior and 2) constructing models based on external power meters.

The formal power modeling method exploits battery behavior to estimate the power consumption within a certain time period. This technique requires the knowledge of the discharging current and remaining battery capacity. There are two representative literature works rely on this technique. In [6], Dong et al. proposed an automatic method that constructs the power model using a smart battery interface. In [7], Gurun et al. proposed an adaptive power estimation model which relies on the built-in Battery Monitor Unit (BMU). The major drawback of these techniques is that the information required by the techniques is quite difficult to obtain. Furthermore, these information is not generally available in most smartphones.

The latter power modeling method relies on external power meter. The external power meter is a hardware based power measurement tool assisted with software logging and visualizing capability. In this technique, power model is constructed by correlating operating system visible state variables with power consumption which is obtained from power meter periodically, while running a set of mobile applications. In [8], Shye et al. proposed a system-level power model for Android based smartphones. The main disadvantage of this approach is that the measurement accuracy highly relies on the training data set which implicitly specifies the relation of system visible state variables with power consumption; therefore, to adopt this approach, the users have to generate as much training data set as possible, and this procedure requires large amount of time and effort.

To overcome above issues, in [4], Zhang et al. proposed a power modeling method named PowerTutor [9]. To the best of our knowledge, PowerTutor is one of the most popular power measurement tools for Android platform, and there are many researchers and developers exploiting this tool to perform power measurement for research and application development purpose. In this work, therefore, we consider the PowerTutor as a baseline approach and compare the proposed solution - PowerGuide with PowerTutor in terms of estimation accuracy on Wi-Fi power consumption. The power model of PowerTutor relies on the knowledge of the battery related metrics that are generally available in most smartphones. The metrics include battery discharge voltage curve and the values obtained from battery voltage sensor. Zhang et al. only considered two parameters (e.g., packet rate and channel rate) to derive the Wi-Fi power model by exchanging fixed size TCP packets between the smartphone and a local server. The PowerTutor suffers from the following two drawbacks:

- **Non-consideration of Inter-Arrival Time:** the PowerTutor does not consider the impact of packet inter-arrival time which significantly alters the Wi-Fi communication pattern. Since energy expenditure is determined by communication pattern, therefore, estimated result by PowerTutor shows low accuracy.
- **Obsolete Power Model:** the PowerTutor was initially introduced in 2010, and the power model and corresponding preconfigured values (e.g., power consumption in each power state) are more tailored to out-of-date smartphones such as HTC Dream and HTC Magic. Through a set of experiments using the recently released smartphones, we found that there was huge discrepancy between estimated power by PowerTutor and measured power by hardware power meter.

With considering these limitations, we design a Wi-Fi power estimation application - PowerGuide by taking into account the packet rate, packet size and packet inter-arrival time. Furthermore, we implement and deploy the PowerGuide to the recently released smartphones for evaluation purpose.

III. Wi-Fi Power Expenditure Characterization

In this section, we investigate key network metrics that influence the Wi-Fi power consumption. To clearly understand how the network metrics contribute to the power consumption, we conducted a set of experiments by varying the value of the considered network metrics.

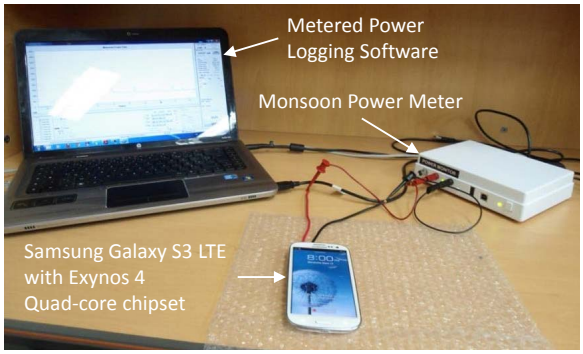
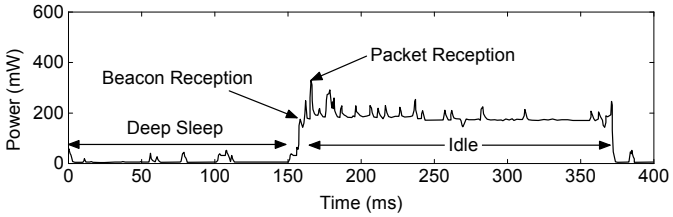


Fig. 1. Wi-Fi power consumption measurement testbed using Monsoon

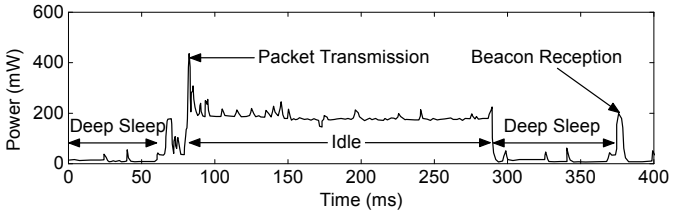
A. Experiment Setup

As shown in Figure 1, our experiment environment is comprised of three main components:

- **Smart Device:** this componet works as Wi-Fi client. We choose the Samsung Galaxy S3 LTE as a Wi-Fi client and the smartphone is running the latest version of Android Jellybean 4.3. The device is equipped with Exynos 4 Quad Core 1.4 GHz CPU, 2 GB RAM and Broadcom's Wi-Fi communication module in which BCM 4334 Wi-Fi chipset [10] is embedded. BCM 4334 is a new Wi-Fi chipset which

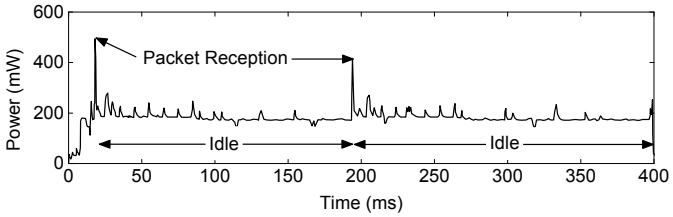


(a) Energy expenditure of downlink case with large packet inter-arrival time

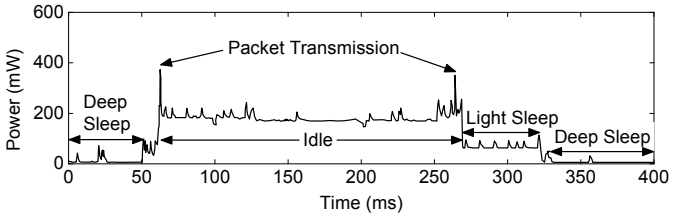


(b) Energy expenditure of uplink case with large packet inter-arrival time

Fig. 2. Energy expenditure with large packet inter-arrival time



(a) Energy expenditure of downlink case with 200 ms inter-arrival time



(b) Energy expenditure of uplink case with 200 ms inter-arrival time

Fig. 3. Energy expenditure with 200 ms inter-arrival time

is widely used in various types of smart devices such as Galaxy Note 2.

- **Power Meter:** we adopt a hardware based power meter - Monsoon [11] to perform power measurement with high accuracy. Monsoon is comprised of two main components that are power monitor (hardware) and power tool (software). Power monitor is directly connected to smartphone's Li-Ion battery. Power tool interacts with power monitor using proprietary protocol through USB serial communication. Power tool is installed in measurement server for logging purpose.
- **Measurement Server:** this component provides two functions: 1) read the measured power from Monsoon using power tool software and store the measured value into files, 2) behave as a virtual wireless Access Point (AP) which accepts the wireless connection from other smart devices through IEEE 802.11 protocol with configurable paremters. Connectify-me is exploited to realize virtual AP.

Since the measurement result by power meter represents the overall power consumption; moreover, the measured value varies from time to time; therefore, it is nontrivial to accurately fetch the Wi-Fi energy consumption only among other hardware devices. We go through following three steps to accurately obtain the Wi-Fi energy consumption in this paper.

1) **Measure the energy consumption through monitoring battery current:** by periodically monitoring the battery supported current, we can obtain the power measurement result with high accuracy using Monsoon.
2) **Filter out energy expenditure by other modules:** existing power measurement tools report the overall energy consumption of all hardware devices rather than individual device, which makes it difficult to obtain Wi-Fi energy consumption only. Among all hardware components, CPU contributes the most in terms of energy consumption, thus, we try to fix the CPU energy consumption to a certain level by throttling CPU frequency under 200 MHz, and then subtract it from the total power consumption. Moreover, we turn off other hardware devices including sensors, display and etc. to mitigate the energy-wide interferences. The refined energy consumption can virtually represent the pure energy consumption by Wi-Fi communication module.
3) **Repeat the above steps by varying network parameters:** we go through above two steps five times and calculate the average and its error rate. We also vary the network metrics such as packet rate, packet inter-arrival time and packet size to figure out how those parameters can affect the Wi-Fi energy consumption.

B. Experiment Study on Wi-Fi Energy Expenditure Pattern

The energy expenditure pattern of Wi-Fi communication module is affected by various network metrics such as transmission direction (e.g., uplink and downlink), packet inter-arrival time, packet size and packet rate. In this subsection, we will show how those network metrics can contribute to Wi-Fi energy consumption.

Figure 2 shows energy expenditure pattern in both data upload and download cases with relatively long packet inter-arrival time. Packet inter-arrival time is a metric which denotes the gap between two consecutive packets. In this experiment, we try to send a single packet with 400 ms inter-arrival time from AP to mobile client and vise versa. As we can observe from experiment result, with large value of packet inter-arrival time regardless of data transmission direction, the amount of energy consumption and its pattern are almost identical. In Deep Sleep (DS) state, the Wi-Fi device spends the least amount of energy, moreover, since the device is not ready for data transmission, it has to wait until the reception of next beacon message. DS state transits to IDLE state through either Packet Reception (PR) state or Packet Transmission (PT) state. Both PR and PT states last only 1 ms for transmitting one packet, and the power state immediately transits to IDLE state. In IDLE state, more energy is required compared to DS state to preserve the transmission readiness. It also means that in IDLE state, regardless of the existence of beacon message, if the device has any data in its buffer space, it will immediately transmit the data.

Although the state transition behaviors are quite similar in both uplink and downlink cases with large inter-arrival time as

shown in Figure 2. However, there will be wide discrepancy on state transition behaviors of two cases, if we lower down the packet inter-arrival time to a certain range. In the second experiment, we repeat the first experiment with same input parameters except using the different packet inter-arrival time value. In downlink case, no matter what inter-arrival time that we choose, the two consecutive packets from AP always generate two surge spikes (RR state), and the RR state transits to IDLE state. With our experiment setup, we found that for single packet transmission, the IDLE state lasts 205 ms, and we use notation T_{IDLE} to denote the duration of IDLE state. The overall transition procedure has been shown in Figure 3(a).

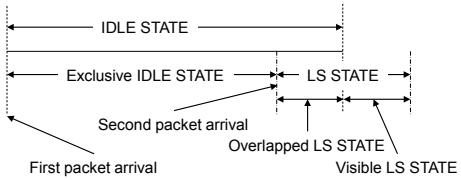


Fig. 4. IDLE and LS state illustration

Unlike to the downlink case, in uplink case, the second packet does not trigger the IDLE state, instead it triggers the Light Sleep (LS) state which consumes less energy than IDLE state but more energy than DS state. However, the duration of LS state is much shorter than that of IDLE state, and the power consumption in LS state is much less than that in IDLE state, therefore, the overlapped LS state is not visible in power meter. From Figure 4, we can observe that the duration of visible LS state varies in accordance with the packet arrival time. To figure out the total duration of LS state, we let the device transmit the second packet right after the IDLE state is terminating. To do so, we observed that LS state lasts 65 ms, and we use notation T_{LS} to denote the duration of LS state (see Figure 3(b)). Therefore, the duration of exclusive IDLE state is in the range $T_{IDLE} \geq t \geq T_{IDLE} - T_{LS}$, and to transit to the visible LS state, the device needs to send the second packet no later than $T_{IDLE} - T_{LS}$ which is 140 ms in our case.

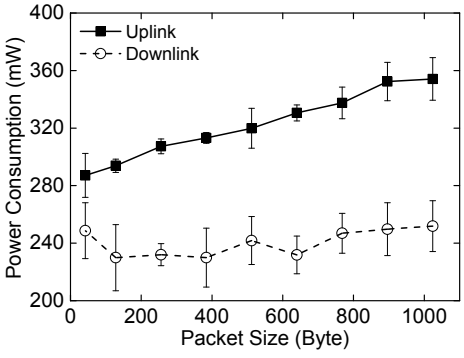


Fig. 5. Power consumption for uplink and downlink with different packet size

In wireless network, the size of the packets is varied in accordance with the type of network workload. For example, the VoIP applications use small size of packets, while FTP applications use much larger size of packet. In following experiment, we would like to show the relation between packet size to energy consumption. The experiment is conducted by varying the size of packet from 10 bytes to 1024 bytes for both uplink and downlink traffic, and the result has been shown in Figure 5. In downlink case, there is no strong correlation between packet size to energy consumption. Regardless of packet size, the energy expenditure rate is all around 240 mW. In uplink case, the energy consumption linearly increases according to the packet size. Such discrepancy is caused by the transmission chain. In downlink case, all received signals must go through the Low Noise Amplifier (LNA) which does not contribute too much on energy expenditure. Whereas, in uplink case, all transmitted signals must go through the Power Amplifier (PA), which consumes energy in per bit manner, therefore, uplink traffic consumes more energy compared to downlink traffic.

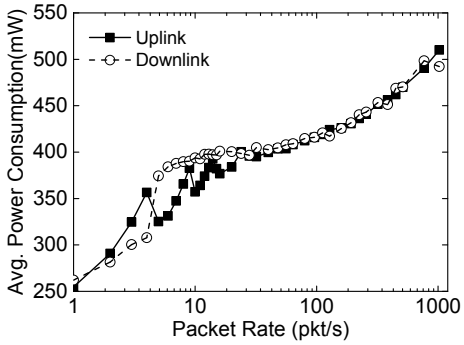


Fig. 6. Power consumption for uplink and downlink with different packet rate

Lastly, we performed a set of experiments to understand how the packet rates can affect the Wi-Fi energy expenditure. We varied the packet rate from 1 pkt/s to 1024 pkt/s, and the data transmission lasts 60 seconds in each packet rate. Each packet is configured to have fixed size and the inter-arrival time is equal to the transmission duration divided by the packet rate. From Figure 6 we can observe that the average energy consumption increases rapidly within packet rate 1 pkt/s to 5 pkt/s, while the increasing trend is relatively gentle when the packet rate over 5 pkt/s. This phenomenon can be interpreted as the fact that for small packet rate the Wi-Fi module can still find the opportunity to go back to the DS state, while for large packet rate, it is difficult or almost impossible to make the Wi-Fi module work in DS state. As the consequence, the energy consumption increases proportionally to the packet rate.

IV. Proposed Wi-Fi Power Model

In this section, we discuss the design of Wi-Fi power model. With considering the observation from previous section, we build our own power model and implement it as a smartphone application named PowerGuide. Our power model is comprised of two sub-models: 1) the energy expenditure rate and duration model for each power state; and 2) the power state transition model.

We derive the energy expenditure rate and duration model by using the experimental observations from Section III. The detailed model design has been shown in Table I. Note that, except the power expenditure ratio in PT state and the time duration in LS state, all other factors are assigned the constants.

TABLE I. ENERGY EXPENDITURE AND DURATION IN POWER STATE

State	Power (mW)	Duration (ms)
PT	$0.069 \times packet\_size + 286$	1
PR	240.0	1
IDLE	200.0	205
LS	80.0	$inter\_arrival\_time - 140$
DS	10.0	N/A

We design our power state transition model based on Finite State Machine (FSM) and the detailed design has been shown in Figure 7. Here $t1$ denotes the total duration of the IDLE state (e.g., 205 ms), whereas $t2$ denotes the duration of exclusive IDLE state (e.g., 140 ms). E denotes the time spent in IDLE state and α denotes a binary variable used to check whether a packet has been transmitted after $t2$. Note that, in Figure 7, we use notation "=" to denote the equality, while use notation "$\leftarrow$" to denote a value assignment.

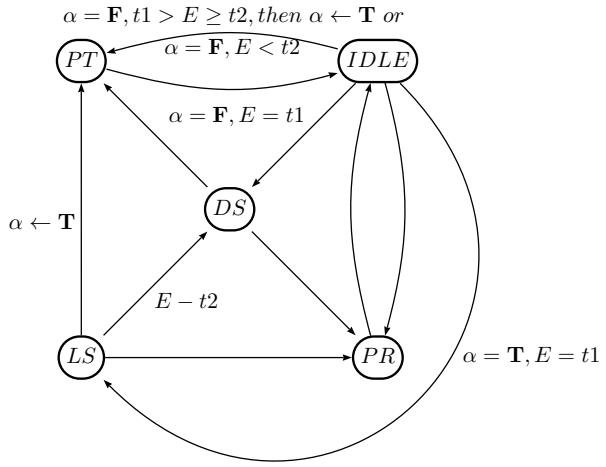


Fig. 7. Power state transition model

The most complex transition is the one transits from IDLE state to other states. There are four potential transitions from IDLE state. The first transition is from IDLE to PT. The value of E cannot exceed $t1$. Therefore, no matter what value that E has, as long as α is **FALSE**, IDLE state always transits to PT. The second transition is from IDLE to DS. This transition occurs when E equals to $t1$ while α is **FALSE**. The third transition is from IDLE to LS. This transition only occurs when α is **TRUE** and E reaches the IDLE state timeout duration. The last transition is from IDLE to PR when it receives a incoming packet from AP. There are two ways to transit from LS to PT. One goes through the DS state in which α will be reset as **FALSE**, while the other one directly goes to PT in which α will still preserve **TRUE** value. By default, the Wi-Fi communication module is in DS state.

By using the proposed power model, we design a Wi-Fi power estimation algorithm, and the pseudo code presentation has been shown in Algorithm 1. The algorithm takes number of transmitted/received packets, each state's power expenditure rate combined with duration constants, and the estimation granularity Γ as input. The final output is the amount of estimated power during Γ. The detailed description on variables has been shown in Table II. Note that, with finer granularity Γ, we can obtain more accurate power estimation result.

TABLE II. VARIABLE USED IN POWER ESTIMATION ALGORITHM

Var. Name	Variable Description
Γ	The granularity of reading the network metrics
N_Γ	The number of packets sent/received during Γ
δ	The duration required to send one packet
$\mathbf{\Psi}_{PT}$	The power spent in the Packet Transmission state
$\mathbf{\Psi}_{PR}$	The power spent in the Packet Received state
$\mathbf{\Psi}_{IDLE}$	The power spent in the IDLE state
$\mathbf{\Psi}_{LS}$	The power spent in the Light Sleep state
$\mathbf{\Psi}_{DS}$	The power spent in the Deep Sleep state
τ_{WAKE}	The time spent in the PT/PR/IDLE states
$AwakeFlag$	A flag denotes whether it is in PT/PR/IDLE state
$LSFlag$	A flag denotes whether it is in LS state
D_{LS}	A duration of LS state
T_{LS}	Elapsed time in LS state
cnt	Number of granularity units elapsed from the last estimation

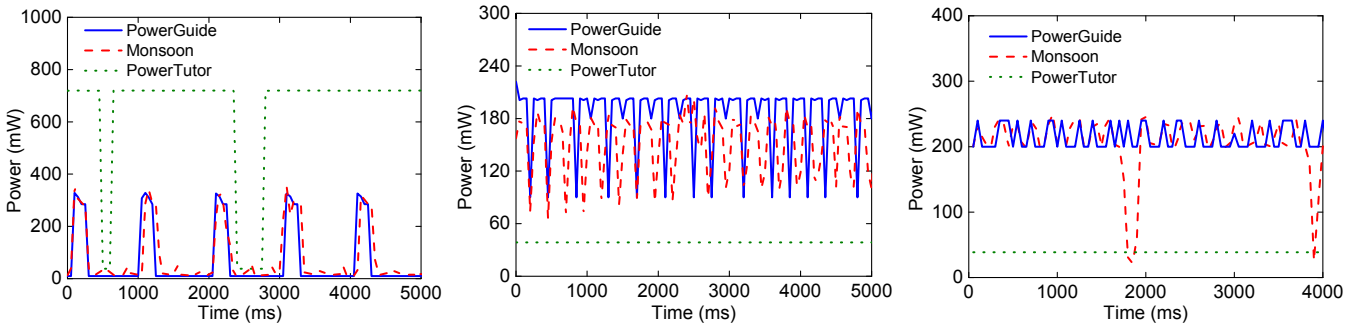
Algorithm 1: Power Estimation Algorithm

input : Γ, $\mathbf{N}_\Gamma$, δ, $\mathbf{\Psi}_{PT}$,$\mathbf{\Psi}_{PR}$, $\mathbf{\Psi}_{IDLE}$, $\mathbf{\Psi}_{LS}$, $\mathbf{\Psi}_{DS}$, τ_{WAKE}
output: Estimated power consumption $\mathbf{\Psi}$

```
while TRUE do
    if N_Γ ≠ 0 then
        if transmitted packets? then
            Ψ ← {Ψ_PT × N_Γ × δ + Ψ_IDLE × (Γ − N_Γ × δ)};
            if AwakeFlag = TRUE then
                LSFlag ← TRUE; T_LS ← D_LS;
        else
            Ψ ← {Ψ_PR × N_Γ × δ + Ψ_IDLE × (Γ − N_Γ × δ)};
        AwakeFlag ← TRUE;
        increment cnt by one;
    else
        if cnt < τ_WAKE/Γ and AwakeFlag then
            Ψ ← Ψ_IDLE;
            increment cnt by one; decrement T_LS by Γ;
        else
            if LSFlag = TRUE and T_LS ≥ 0 then
                Ψ ← Ψ_LS; decrement T_LS by Γ;
            else
                Ψ ← Ψ_DS;
            AwakeFlag ← FLASE; cnt ← 0;
    wait Γ;
```

V. EVALUATION

To validate our work, we deployed the PowerGuide into the smartphone to estimate the Wi-Fi power consumption. We used the same experiment setup introduced in Section III. We chose the PowerTutor as the baseline method, and measurement result from Monsoon as the ground truth. Since the PowerTutor only relies on the packet rate in which 8 pkt/s is the PSM and CAM modes' switching threshold; hence, to prove the inaccuracy of PowerTutor's power model we sent the packets from mobile devices with 9 pkt/s and 4 pkt/s packet rate in the first and the second experiment, respectively. Due to the trade-off relation between power estimation accuracy and power conservation, we used the value of 50 ms as the estimation granularity Γ in the following experiments.



(a) Comparison result on estimated power consumption for uplink traffic with 9 pkt/s packet rate, 6 ms inter-arrival time, and 1KB packet size

(b) Comparison result on estimated power consumption for uplink with 4 pkt/s packet rate, 250 ms inter-arrival time, and 1KB packet size

(c) Comparison result on estimated power consumption for downlink with 5 pkt/s packet rate, 200 ms inter-arrival time, and 1KB packet size

Fig. 8. Comparison result on estimated power consumption using PowerGuide and PowerTutor

Figure 8(a) shows the power estimation result with 9 pkt/s packet rate and 6 ms packet inter-arrival time. With considering the IDLE state duration (e.g., 205 ms), the device worked in active state (IDLE + PT) around 259 ms per second. However, PowerTutor only took into account packet rate for power estimation, hence, it made wrong assumption such that the device was almost fully working in CAM. Whereas, PowerGuide was able to estimate the power consumption accurately with very minor error rate. Overall, the estimation error rate of PowerTutor was around 1200%, while that of PowerGuide was around 14%.

Figure 8(b) shows the power estimation result with 4 pkt/s packet rate and 250 ms packet inter-arrival time. With this parameter setup, we observed evenly spread packet transmission pattern, and each packet induced around 205 ms IDLE state and 45 ms DS state. Since 45 ms is quite small number, the Wi-Fi device could not find much opportunity to go back to the DS state, thus, we cannot observe clear DS state in this experiment. PowerTutor underestimated the power consumption, as it assumed that the device was fully working in PSM (DS state). Overall, the error rate of PowerTutor was around 74%, while that of PowerGuide shown 15% error rate which was around 5 times more accurate compared to the result obtained from PowerTutor.

In the third experiment, we performed power estimation on uplink traffic with 5 pkt/s packet rate and 200 ms packet inter-arrival time, and Figure 8(c) shows the result. Again, PowerGuide shown very high estimation accuracy compared to PowerTutor. We observed minor estimation miss match on power state using PowerGuide, and this was caused by large estimation granularity which was configured as 50 ms in this experiment. Overall, in above three experiments, we achieved around 86% power estimation accuracy using PowerGuide.

VI. Conclusion

In this paper, we presented PowerGuide - an accurate Wi-Fi power estimation tool for smartphones. We characterized the Wi-Fi energy consumption pattern considering packet rate, packet size and packet inter-arrival time. Based on the experimental study, we designed a power model for accurately estimating the Wi-Fi energy consumption by smartphones. The proposed power model was based on FSM and was implemented as a Wi-Fi power estimation application targeted to Android OS. PowerGuide outperformed PowerTutor with an overall estimation accuracy around 86%.

Although the current implementation of PowerGuide was only targeted to BCM 4334 Wi-Fi chipset, there will not be significant difference between BCM 4334 to the other Wi-Fi chipsets in terms of FSM model. This is because, most of the existing Wi-Fi chipsets all rely on IEEE 802.11 protocol which characterizes the transmission pattern. Therefore, as a future work, we will further seek a way to automate the procedure of obtaining the power model constants.

References

[1] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proceedings of the 2010 USENIX conference on USENIX annual technical conference (USENIXATC 2010)*, 2010.

[2] A. Pathak *et al.*, "What is keeping my phone awake?: Characterizing and detecting no-sleep energy bugs in smartphone apps," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys '12)*, 2012, pp. 267–280.

[3] X. Chen, Y. Chen, Z. Ma, and F. C. A. Fernandes, "How is energy consumed in smartphone display applications?" in *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications HotMobile '13*, 2013, pp. 3:1–3:6.

[4] L. Zhang *et al.*, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, 2010, pp. 105–114.

[5] R. Mittal, A. Kansal, and R. Chandra, "Empowering developers to estimate app energy consumption," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking (MobiCom '12)*, 2012, pp. 317–328.

[6] M. Dong and L. Zhong, "Self-constructive high-rate system energy modeling for battery-powered mobile systems," in *Proceedings of the 9th international conference on Mobile systems, applications, and services (MobiSys '11)*, 2011, pp. 335–348.

[7] S. Gurun and C. Krintz, "A run-time, feedback-based energy estimation model for embedded devices," in *Proceedings of the 4th international conference on Hardware/software codesign and system synthesis*, 2006, pp. 28–33.

[8] A. Shye, B. Scholbrock, and G. Memik, "Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2009, pp. 168–178.

[9] "PowerTutor," http://powertutor.org/.

[10] "BCM4334 Wi-Fi Chipset," https://www.broadcom.com/products/Wireless-LAN/802.11-Wireless-LAN-Solutions/BCM4334/.

[11] "Monsoon Power Monitor," http://msoon.com/.