FSM-based Wi-Fi Power Estimation Method for Smart Devices

Jian Li*, Jin Xiao†, 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

***Abstract*—With the increased popularity of mobile data applications, Wi-Fi power consumption on smartphones is now a significant portion of mobile energy expenditure. In their efforts to develop more energy efficient applications, the applications developers use energy estimation tools as a benchmark. Although hardware based power meter has high estimation accuracy, it is cumbersome to operate as it relies on physically attaching wires to the battery, and moreover the price of hardware based power meter is expensive. Therefore software based power estimation tools such as PowerTutor are popular. In our prior research using PowerTutor as power meter, we discovered that PowerTutor has large Wi-Fi power estimation error (over 1000%) on post 2012 phones. In this work, we propose a new FSM-based Wi-Fi power model based on IEEE 802.11 communication patterns and Wi-Fi hardware configuration with significantly increased power estimation accuracy. We designed and implemented our power model as an estimation tool called PowerGuide. Our experiments with PowerGuide in field operations showed that PowerGuide can achieve an average estimation accuracy of 86% compared to hardware power meters even with moderate polling period.**

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

I. Introduction

With the unstoppable growth of smartphones, mobile applications are popular more than ever. Among the overall mobile applications, the networked mobile applications take the largest proportion, and rely on Wi-Fi communication technology for efficiency and economic reasons. Consequently, Wi-Fi power consumption is contributing to a significant portion of modern smartphone energy usage [1]. Understanding this trend, smartphone Wi-Fi standardization groups have made much effort in providing tunable options for application developers to design and implement their applications in more energy efficient ways. Application developers often times use energy profiling tools to benchmark their applications, unfortunately many existing methods such as [2][3][4][5][6] only profile the power usages per smartphone application, based on CPU and/or display consumptions, not Wi-Fi or communication. In this context, this paper studies the problem of accurate Wi-Fi power monitoring for end users and application developers. Hardware based power meters provide accurate power measurement, but they are expensive and impractical for regular use due to the need to physically attache wires to the battery. Therefore, software based power estimators are becoming popular. Existing software based power estimation tools (such as the popular PowerTutor [7], and research work in [8][9][10]) have low measurement accuracy due to their simplified power model assumptions and lack of consideration for modern smartphone Wi-Fi hardware operation patterns. In general, Wi-Fi power estimation tools rely on a specific power model derived from 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 controlled 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 power consumption.

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 also the packet inter-arrival time which the existing power estimation tools do not consider; and 3) the uplink and downlink power expenditure show asymmetric characteristics while existing power estimation tools assume symmetric behavior. These discrepancies resulted in existing power estimation tools showing low estimation accuracy especially on contemporary smartphones. Accordingly, we proposed a new power model and developed a new power estimation tool called PowerGuide. To understand the key design elements of our model, we first performed detailed experimental studies to characterize the power patterns and evaluated the significance of different network metrics on power consumption. Our resulting Finite State Machine (FSM) based power model considers new network metrics that are not considered in other power estimation tools and is novel in capturing the different Wi-Fi communication states and their transitional conditions. We implemented the power model as a mobile application for Android called PowerGuide. We validated PowerGuide by deploying it on the latest Android smartphones, conducted field experiments under diverse settings, and compared our estimation result with both PowerTutor and hardware power meter. Our results showed that PowerGuide can achieve an average estimation accuracy of 86% compared to hardware power meters even with moderate polling period, and is very easy to use and versatile for 802.11n smartphones.

978-3-901882-50-0

The remainder of this paper is organized as follows. Section II presents related works on Wi-Fi power estimation. With a hardware power monitoring tool, we study Wi-Fi power consumption pattern under different network conditions in Section III. In Section IV, we derive the FSM-based power model, and present the implementation of PowerGuide in Section V. Section VI reports the result of our field studies. Section VII conclude the paper with a description of future work.

II. Related work

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

The former power modeling method estimates power consumption based on battery behavior within a certain time period. This technique requires knowing the discharging current and remaining battery capacity. Here, we summarize three representative works following this technique. In [11], Dong et al. proposed an automatic method that constructs the power model using a smart battery interface. In [12], Gurun et al. proposed an adaptive power estimation model based on the built-in Battery Monitor Unit (BMU). The method's accuracy is strongly influenced by the update rate of BMU, with low update rate resulting in low power estimation accuracy. To alleviate this limitation, Jung and Yoon et al. proposed DevScope [13] and AppScope [14] in which hardware components are configured based on the BMU update rate. Overall, battery behavior based methods rely on accurate and frequent battery statistics which are difficult to obtain in most smartphones.

The later method uses a hardware based power meter accompanied with software logging and visualizing tools. With this technique, power model is constructed by correlating operating system visible state variables with power consumption periodically obtained from power meter readout, profiled against a set of mobile applications. In [15], Shye et al. proposed a system-level power model for Android based smartphones, and in [16], Sun et al. proposed a Wi-Fi power-throughput model based on linear and non-linear statistic modeling. This technique in general requires large training data set that can well capture the relations between system visible state and power consumption. The search space is also very large given the highly varied types of mobile applications as well as communication chips out on the market today.

To overcome above issues, in [7], Zhang et al. proposed a power estimation tool named PowerTutor [17]. To date, PowerTutor is perhaps the most popular power measurement tools for Android platform. PowerTutor uses a power model to estimate smartphone power consumption. The Wi-Fi power model of PowerTutor includes a state transition model which reflects the communication pattern, and the power consumption rate in each state. To build the power model, they collected several power metrics as well as communication metrics which were easily obtainable from recent smartphones, correlated the two types of metrics, and finally resulted a state transition model. However the considered communication metrics are restricted to packet rate and channel rate only, which cannot correctly reflect the IEEE 802.11 communication pattern. We summarize the three major issues of PowerTutor as follow:

- **Symmetric Power Model for Uplink and Downlink:** PowerTutor adopted the identical power model for both uplink and downlink cases. However, due to the discrepancy of communication pattern of uplink and downlink, the power model for each case is different.
- **Non-consideration of Packet Inter-Arrival Time:** Packet inter-arrival time is a key metric in determining the Wi-Fi communication pattern. By not considering packet inter-arrival time, PowerTutor shows low accuracy in practice.
- **Obsolete Power Model:** The power model was derived in stochastic manner rather than careful consideration of the IEEE 802.11 communication pattern and behavior, therefore it only supports the phones used in power model design period. Since this research was conducted in 2009, as a consequence, the power model only supports old fashion phones such as HTC Dream and HTC Magic.

Our approach is similar to the PowerTutor except we take into account more communication metrics including the transmission direction, packet size and packet inter-arrival time to better capture the IEEE 802.11 communication pattern, and hence, results higher Wi-Fi power estimation accuracy. Our power model can tackle the issue of new smart device evolution at the fundamental level, because the upcoming Wi-Fi devices implement the IEEE 802.11 standard that our power model captures. We are able to show that our model works for new upcoming smart devices with simple parameter tuning. In fact, we developed our model on Samsung Galaxy 3, and our experiment show that the model is valid for the new Samsung Galaxy 5 as well which used a different communication chip.

III. Wi-Fi Power Expenditure Characterization

In this section, we identify important Wi-Fi network metrics that affect the Wi-Fi power consumption, from a set of relevant network metrics: Received Signal Strength Indicator (RSSI), transmission direction (e.g., uplink and downlink), packet size, packet rate and packet inter-arrival time. Our methodology is to conduct sets of controlled experiments varying each of the considered network metric. Variability in measurement noise are smoothed out by conducting each experiment setting repeatedly until a statistical significant central value can be observed, or correlation is ruled out in case no central value is present.

A. Experiment Setup

As shown in Figure 1, our experimental 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 which uses BCM4334 Wi-Fi chipset [18] operated in 802.11n mode. A Li-Ion battery with 2,100 mAh capacity is equipped inside the smartphone. For experiment control purpose, an UDP packet sender/receiver application with a set of configurable parameters is developed and installed on the smartphone.

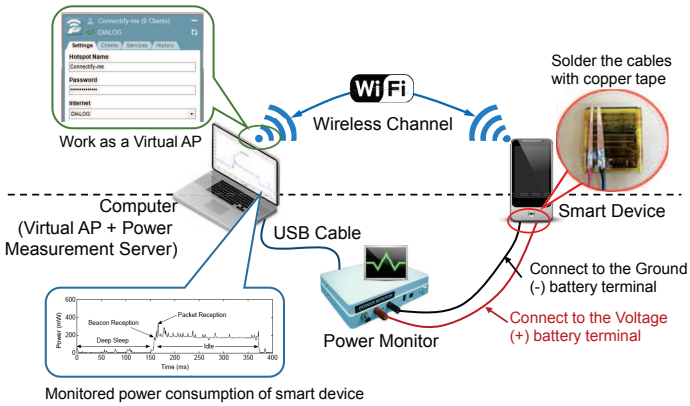


Fig. 1. Experiment setup for studying the relations between network metrics and power consumption

- **Power Monitor:** We acquired a hardware based power meter - Monsoon [19] to perform power measurement. Monsoon has two sub-components: the power monitor (hardware) and a power tool (software). Power monitor has red and black terminals which are connected to the voltage (+) battery terminal and ground (-) battery terminal respectively. The power monitor periodically monitors the current draw of smartphone, and the monitored results are constantly reported to power tool using proprietary protocol through USB serial communication. Power tool is installed into a commodity computer (measurement server) for logging and visualization purpose.
- **Measurement Server:** This component provides three functions: 1) read the measured result from power monitor using power tool, log the measured result into the files and visualize it; 2) behave as a wireless Access Point (AP) which accepts the wireless connection from the smart devices through IEEE 802.11 protocol. Connectify-Me [20] is used to turn the server into a virtual wireless AP; 3) send/receive the UDP packet to/from smart devices.

The measurement result obtained by power meter represents the total power consumption of the smartphone. It is nontrivial to accurately obtain the Wi-Fi power consumption portion only. We used the following two-step process to extract the Wi-Fi power consumption portion that is reported in this paper.

1) **Measure the overall power consumption:** Since the battery current flows as the smart device consumes the power, by periodically monitoring the current flow from battery, the meter can very accurately obtain power consumption information.
2) **Filter out power consumption by other hardwares:** The obtained power consumption is the total power consumption. We eliminate the power consumption of non Wi-Fi hardware components in the following ways: 1) we fix the CPU power consumption to a certain amount by underclocking CPU frequency down to 200 MHz using SetCPU app [21], and then subtract it from the total energy amount; 2) we turn off hardware modules (e.g., sensors, display, GPS, bluetooth, etc.); and 3) we adopt a process manager to terminate all irrelevant background processes which would possibly contaminate the measurement result. Finally, the leftover power consumption can largely be attributed to Wi-Fi with minor noise.

The above process is used to conduct sets of experiments by varying the investigated network metrics and experiment multiple times to achieve statistical significance.

B. RSSI

RSSI quantifies the power level present in a received radio signal. RSSI typically has inverse relation with the distance between Wi-Fi client and AP. With low RSSI value, the data loss and error rate increase. Therefore, with large distance, the Wi-Fi client tries to increase the transmission power in order to lower down the data loss, which in turn increases Wi-Fi power consumption in smart devices. To figure out the specific impact of RSSI on power consumption, we varied the RSSI value by placing the smart device in different location, and measured the Wi-Fi power consumption using Monsoon. Figure 2 shows the observed results between RSSI and power consumption. The scatters represent the power consumption, while the solid line denotes the fitting curve. We do not observe significant correlation between RSSI and Wi-Fi power consumption within typical RSSI range (-10 dBm $\sim$ -70 dBm). Therefore, it is infeasible to use RSSI network metric to deduce the Wi-Fi power consumption correctly. Similar result has also been reported in the literature [16].

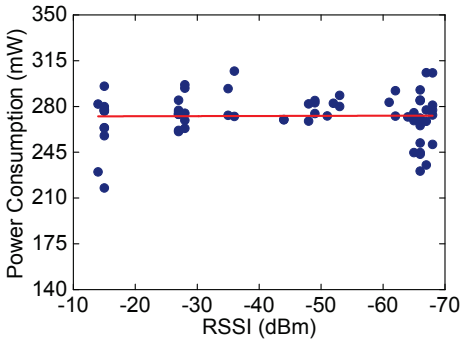
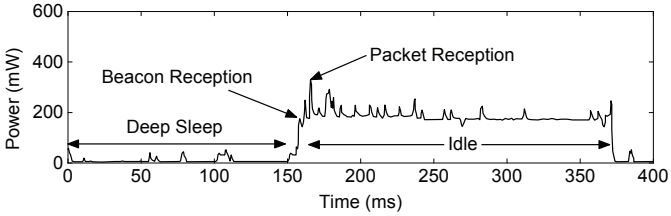


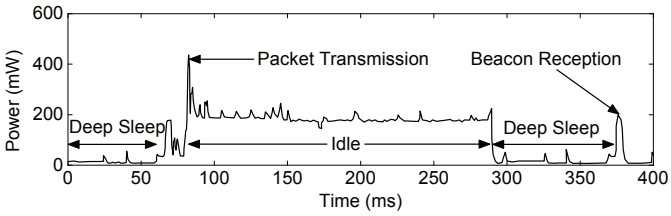
Fig. 2. Power consumption within different RSSI range

C. Packet Transmission Direction and Inter-arrival Time

Figure 3 shows power consumption pattern in both data upload and download cases with relatively long packet inter-arrival time. Packet inter-arrival time is a metric representing the time elapsed between two consecutively transmitted packets. In this experiment, we try to send a single packet with 400 ms inter-arrival time from AP to mobile client and vice versa. As we can observe from experiment result, with large value of packet inter-arrival time regardless of data transmission direction, the power consumption pattern is 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

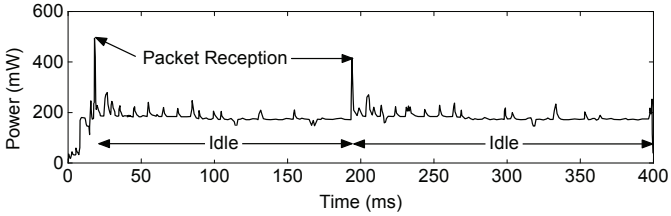


(a) Power consumption of downlink case with large packet inter-arrival time

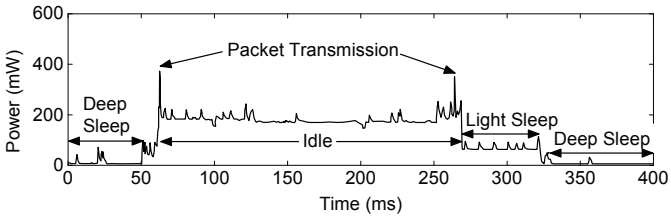


(b) Power consumption of uplink case with large packet inter-arrival time

Fig. 3. Power consumption with large packet inter-arrival time



(a) Power consumption of downlink case with 200 ms inter-arrival time



(b) Power consumption of uplink case with 200 ms inter-arrival time

Fig. 4. Power consumption with 200 ms inter-arrival time

existence of beacon message, if the device has any data in its buffer space, it will immediately transmit the data.

With large packet inter-arrival time, the state transition behaviors are quite similar in both uplink and downlink cases as shown in Figure 3. However, when packet inter-arrival time is smaller, we observe very different behaviors between the uplink and downlink cases. In the second experiment set, we use the same input parameters as the first experiment set, by varying the packet inter-arrival time. In the downlink case, no matter what inter-arrival times are choosen, the two consecutive packets from AP always generate two surge spikes (PR state), and the PR 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 is depicted in Figure 4(a).

In the case of uplink transmission, we observe that upon transmission of the first packet, regardless when the subsequent packet is sent, the time gap between second transmission to the start point of "DS" state is constant (approx. 65 ms). We deduce that this constant time gap represents another power state, which we termed it Light Sleep (LS) state. PowerTutor observed similar occurrence which they termed "ltransmit" [7].

Our observed LS state transition conditions are as follows: for uplink, the second packet triggers LS state and the power consumption in LS state is much less than in IDLE state. The duration of the visible LS state varies according to packet inter-arrival time, whereas the length of the total LS state is the same. We illustrate this relationship in Figure 5. From the figure, we see that part of the LS state duration is masked by the IDLE state (power measurements indicate that IDLE state is the active state). 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 4(b)). Therefore, the duration of exclusive IDLE state is in the range $T_{IDLE} \geq t \geq T_{IDLE} - T_{LS}$, and in order to transit to the visible LS state, the device has to send the second packet no later than $T_{IDLE} - T_{LS}$ which is 140 ms in our case.

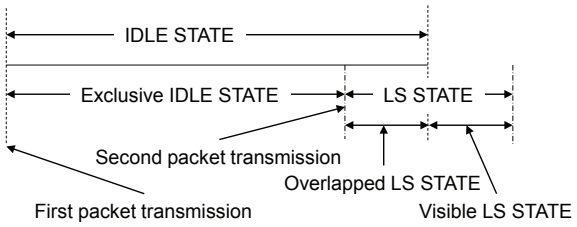


Fig. 5. IDLE and LS state illustration

D. Packet Size

In Wi-Fi network, the size of a packet is varied depending on the type of application. For example, packets of VoIP applications are typically small, while file transfer applications (e.g., FTP) has large packet size. Therefore, it is important to investigate whether packet size has non-negligible impact on Wi-Fi power consumption. In order to correctly control the packet size, we need to ensure that each packet size should not exceed the Maximum Transmission Unit (MTU), otherwise the packet would be divided into several IP fragments, this in turn alters the packet size. To avoid IP fragmentation, we configure the MTU to a rather large size, 1400 bytes in following experiments. The experiment is conducted by varying the size of packet from 10 bytes to 1024 bytes for both uplink and downlink traffic. Two sets of experiments are conducted for both uplink and downlink workload, and the results are shown in Figure 6. In downlink case, there is no strong correlation between packet size to power consumption. Regardless of packet size, the power consumption rate is all around 240 mW. In uplink case, the power consumption linearly increases according to the packet size. Such discrepancy is caused by the transmission chain. In downlink case, all received signals go through Low Noise Amplifier (LNA) which is very power light. Whereas, in uplink case, all transmitted signals must go through the Power Amplifier (PA), which consumes power in per bit manner, therefore, uplink traffic consumes more power compared to downlink traffic [22].

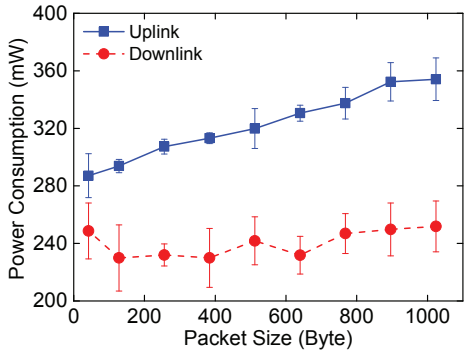


Fig. 6. Power consumption for uplink and downlink traffic with different packet size

E. Packet Rate

Lastly, we performed a set of experiments to understand how the packet rates affect the Wi-Fi power consumption. 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 7 we can observe that the average power consumption increases rapidly within packet rate 1 pkt/s to 5 pkt/s, while the slope is much gentler when the packet rate is over 5 pkt/s. We believe this happens because for small packet rate the Wi-Fi module can still go back to the DS state periodically, while for large packet rate, the Wi-Fi module does not go to DS state at all. Consequentially, although power consumption increases proportionally to the packet rate, we see the two different slopes.

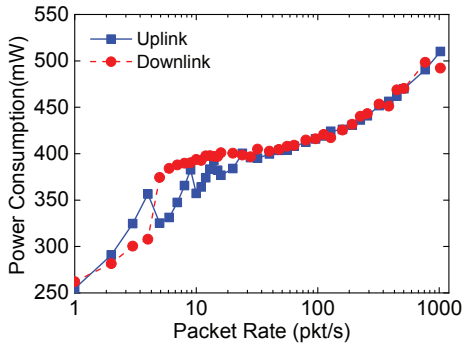


Fig. 7. Power consumption for uplink and downlink traffic with different packet rate

IV. FSM-Based Wi-Fi Power Model

Our experiments as reported in Section III taught us two lessons. Lesson 1, direction of transmission, packet inter-arrival rate, packet size and packet rate are important factors to consider when modeling Wi-Fi power consumption. Lesson 2, the specific state the communication module is in (i.e., PT, IDLE, DS, LS and PR) and the transition conditions between each state, are important modeling considerations. Accordingly, in this section we present the design of our Finite State Machine (FSM) based Wi-Fi power model. Our power model is comprised of two sub-models: 1) the power consumption rate and duration model for each power state; and 2) the power state transition model.

TABLE I. Power consumption 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 derive the power consumption rate and duration model by using the experimental observations from Section III. The detailed model design has been shown in Table I. We observe that except the power expenditure ratio in PT state and the time duration in LS state, all other factors have constant parameters.

We design our power state transition model as a FSM shown in Figure 8. t_1 denotes the total duration of the IDLE state (e.g., 205 ms), whereas t_2 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 t_2. Note that, in Figure 8, we use notation "=" to denote the equality, use "←" to denote a value assignment.

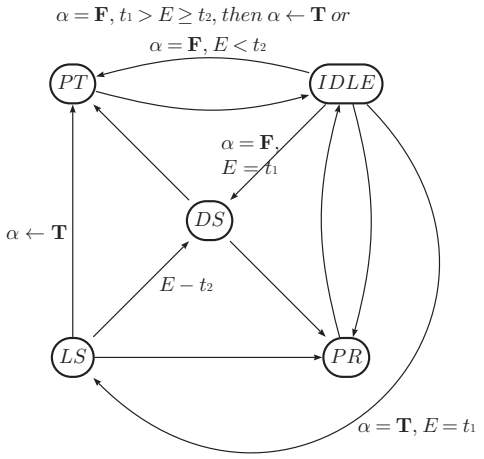


Fig. 8. Power state transition model

The most complex transition is the one 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 t_1. 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 t_1 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 an 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 to **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.

Consequently, we can calculate the overall Wi-Fi power consumption by using the Equation 1.

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
Ψ_{PT}	The power spent in the Packet Transmission state
Ψ_{PR}	The power spent in the Packet Received state
Ψ_{IDLE}	The power spent in the IDLE state
Ψ_{LS}	The power spent in the Light Sleep state
Ψ_{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

$$\Psi_{avg} = \frac{\sum_{s \in \mathbb{S}'} \Psi_s \times \mathbf{T}_s + (\sigma \times \gamma + \Psi'_{PT}) \times \mathbf{T}_{PT}}{\mathbf{T}} \quad (1)$$

where $\mathbb{S}' = \{PR, IDLE, LS, DS\}, \mathbb{S} = \mathbb{S}' \cup \{PT\}$,
σ is power increasing constant with packet size γ,
$\mathbf{T} = \sum_{s \in \mathbb{S}} \mathbf{T}_s$, Ψ'_{PT} is the base power consumption in PT.

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

Algorithm 1: Power Estimation Algorithm

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

```
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. DESIGN AND IMPLEMENTATION OF POWERGUIDE

We designed and implemented our Wi-Fi power model and power estimation algorithm in a smartphone application called PowerGuide. In order to implement our power model, the input network metrics should be available to PowerGuide which runs in user space. However, some network metrics such as packet size, packet inter-arrival time can only be accessible from the kernel space, and moreover those metrics are not exposed by default in the newer Android OS versions. Hence, we implemented a set of kernel modules to make the network metrics available to the application via "/sys" interface [23]. "/sys" is a special virtual filesystem, which stores the information and statistics about kernel subsystems and physical devices including Wi-Fi communication module. The detailed implementation architecture is depicted in Figure 9.

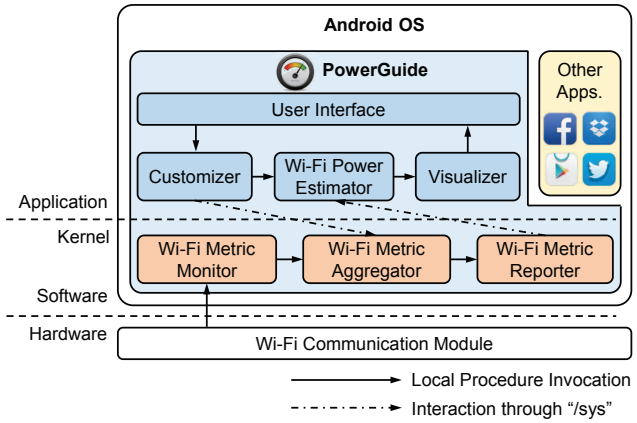


Fig. 9. Implementation Architecture of PowerGuide

- **User Interface:** Through this component, the user can configure the granularity of monitoring network metrics, and also can view the Wi-Fi power consumption pattern in real time.
- **Customizer:** This component is in charge of quantifying the user input signal such as monitoring granularity, and delivering the quantified value to Power Estimator and Metric Aggregator.
- **Power Estimator:** Power Estimator obtains the monitoring granularity as well as the Wi-Fi metric information as input, and calculate the estimated Wi-Fi power consumption by using the Wi-Fi power model proposed in Section IV. Note that, the Wi-Fi metrics get polled periodically by the Power Estimator from "/sys", and therefore, more frequent polling generates more accurate estimation result.
- **Visualizer:** This component fetches the estimated power consumption from Power Estimator, and visualizes it into 2D graph. Visualizer component is embedded into the User Interface to visualize the Wi-Fi power consumption pattern.

- **Kernel Modules:** There are three Kernel Modules developed to assist with the components in application level, and those are Metric Monitor, Metric Aggregator and Metric Reporter. Metric Monitor directly reads the Wi-Fi metrics from the Wi-Fi communication module, and periodically transfers the metrics to the Metric Aggregator. Metric Aggregator aggregates the Wi-Fi metrics with a certain periodicity preconfigured by user. Finally the aggregated Wi-Fi metrics are reported to Wi-Fi Power Estimator through "/sys" interface. With current implementation, the Kernel Modules cannot spontaneously report the fresh Wi-Fi metrics to application components, hence, the application components might use the out-of-date metric for a while.

VI. Evaluation

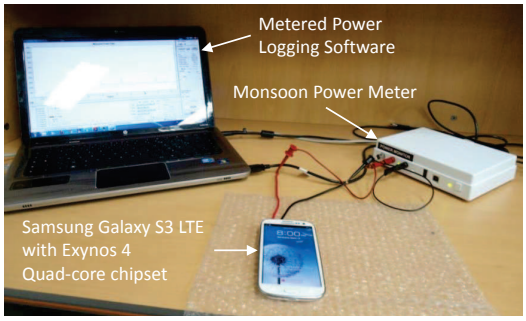


Fig. 10. Wi-Fi power consumption measurement testbed using Monsoon

To test our power model and PowerGuide application in practice, we deployed the PowerGuide and PowerTutor into the Galaxy S3 LTE to estimate the Wi-Fi power consumption. We re-utilized the physical setup shown in Section III, and used Monsoon to accurately measure the Wi-Fi power consumption. We adopted the resulting value from Monsoon as the ground truth, while the estimated power consumption from PowerTutor as a comparison point. So the power estimation tool, whose results are more proximate to the result of Monsoon, will be evaluated to have higher estimation accuracy. In order to better evaluate our power model under specific network conditions, we used UDP traffic in the experiments. We note that PowerGuide works equally well with TCP traffic in practice. But from evaluation point of view, the results obtained under TCP are difficult to interpret/illustrate since the various TCP protocol specifics (ACK messages, transmission window, etc.) introduce variations in key network parameters that drive power state transitions and impacts power consumption. The overall testbed is shown in Figure 10. Four sets of experiments are carried out. In each experiment, the Wi-Fi power consumption is reported by Monsoon, PowerGuide and PowerTutor. Since the PowerTutor only relies on the packet rate in which 8 pkt/s is the PSM and CAM modes' switching threshold, we sent the packets from mobile devices with 9 pkt/s and 4 pkt/s packet rate in the first and the second experiment sets respectively, while we configured the mobile devices to received the packets which had 9 pkt/s and 5 pkt/s packet rate in the third and the forth set of experiments. Due to the trade-off between power estimation accuracy and power conservation, we used the value of 50 ms as the estimation granularity Γ in all experiments.

Figure 11(a) shows the power estimation result with 9 pkt/s packet rate and 6 ms packet inter-arrival time on uplink traffic. 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 assumed that the device was almost fully working in CAM. Whereas, PowerGuide was able to estimate the power consumption more accurately. Overall, the estimation error rate of PowerTutor was around 1,067%, while that of PowerGuide was around 15%.

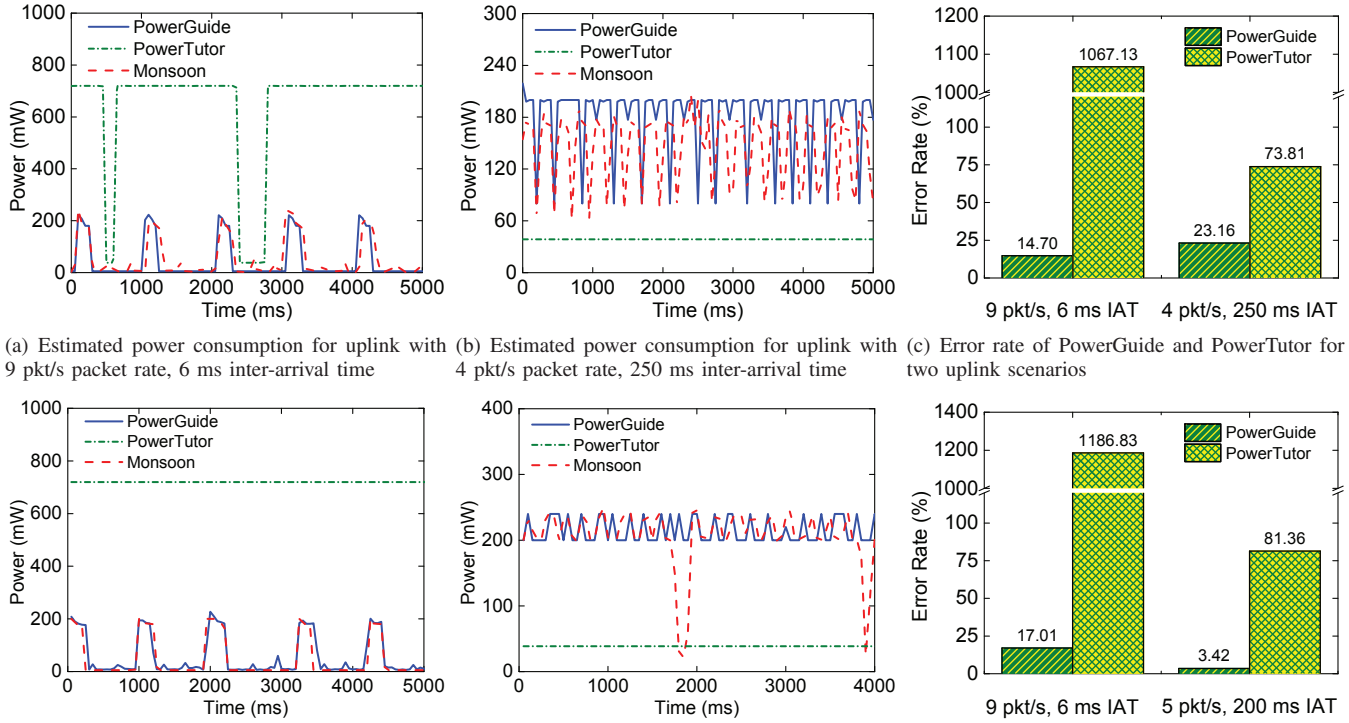
Figure 11(b) shows the power estimation result with 4 pkt/s packet rate and 250 ms packet inter-arrival time. With this parameter setup, we observed evenly spread packet transmission pattern, and each packet induced around 205 ms IDLE state and 45 ms DS state. Since 45 ms is quite short, the Wi-Fi device 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 was 23% that is around 3.5 times more accurate compared to the result obtained from PowerTutor. The error rate of estimated power on uplink traffic is depicted in Figure 11(c), PowerGuide significantly outperforms PowerTutor in all cases.

In the third experiment, we performed Wi-Fi power estimation on downlink traffic with 9 pkt/s packet rate and 6 ms packet inter-arrival time parameter setup, and the result is reported in Figure 11(d). Similar to the trend in Figure 11(a), PowerTutor has very high error rate due to the deficiencies in its power model, while PowerGuide using our FSM-based power model is able to accurately capture the state transitions of the communication module. As a result, PowerGuide showed around 17% error rate compared to PowerTutor's 1,186% in this experiment.

In the fourth experiment, we performed Wi-Fi power estimation on downlink traffic with 5 pkt/s packet rate and 200 ms packet inter-arrival time, and Figure 11(e) shows the result. Again, PowerGuide achieved very high estimation accuracy compared to PowerTutor. We observed small estimation miss match on power state using PowerGuide at around 2,000 ms and 4,000 ms, and this is attributed to our chosen estimation granularity (50 ms) missed out on the short transitions.

Interestingly, we observe that the error rate of the second experiment is much higher than that of the fourth experiment. This is because in the fourth experiment, we chose a smaller packet inter-arrival time of 200 ms, which forced the Wi-Fi device to be awake more frequently compared to the case when we chose the packet inter-arrival time to be 250 ms. So with smaller packet inter-arrival time of 200 ms, the power state was interchangeable only among AWAKE (e.g., PT, PR or IDLE) states and the power state transition frequency was relatively large. As a consequence, even with the relatively coarse estimation granularity of 50 ms, PowerGuide can more accurately detect the power state of Wi-Fi device, and this in turn increases the power estimation accuracy.

The overall error rate of PowerGuide was around 15%, and was mainly due to the polling frequency. In our current implementation of PowerGuide, the Power Estimator periodically polls the latest Wi-Fi metrics stored in "/sys", therefore, even



(a) Estimated power consumption for uplink with 9 pkt/s packet rate, 6 ms inter-arrival time

(b) Estimated power consumption for uplink with 4 pkt/s packet rate, 250 ms inter-arrival time

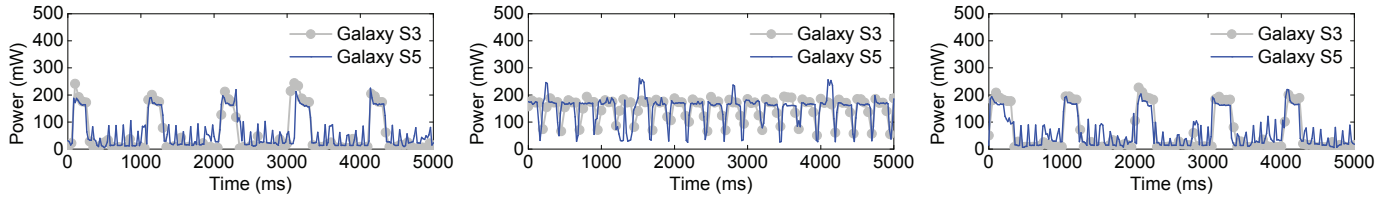
(c) Error rate of PowerGuide and PowerTutor for two uplink scenarios

(d) Estimated power consumption for downlink with 9 pkt/s packet rate, 6 ms inter-arrival time

(e) Estimated power consumption for downlink with 5 pkt/s packet rate, 200 ms inter-arrival time

(f) Error rate of PowerGuide and PowerTutor for two downlink scenarios

Fig. 11. Uplink and downlink estimated Wi-Fi power consumption using PowerGuide and PowerTutor



(a) Power consumption for uplink traffic with 9 pkt/s packet rate, 6 ms inter-arrival time

(b) Power consumption for uplink traffic with 4 pkt/s packet rate, 250 ms inter-arrival time

(c) Power consumption for downlink traffic with 9 pkt/s packet rate, 6 ms inter-arrival time

Fig. 12. The uplink and downlink Wi-Fi power consumption of Galaxy S3 and S5 using Monsoon

if we have fresher Wi-Fi values reported in the kernel, before the next polling period, we cannot obtain it in the application, and this reduces estimation accuracy. We are investigating a way to inform and deliver updated values to application more rapidly from the kernel space. And in doing so, we can further improve accuracy. We intent to investigate this direction in a future work.

Although the current implementation of PowerGuide was only targeted to BCM4334 Wi-Fi chipset, we want to ascertain that our FSM-based power model is a viable general approach, which will not be rendered obsolete with newer generation of smartphones. We reasoned that existing and near future Wi-Fi chipsets all follow IEEE 802.11 standard protocol which is in essence what our FSM captures. To validate this reasoning, we performed the power measurement for Samsung Galaxy 3 LTE (SHV-E210S) and Samsung Galaxy 5 LTE-A (SM-G900K) smartphones using Monsoon power meter. The S5 is equipped with BCM4354 Wi-Fi chipset [24], and it runs the latest version of Android Kitkat 4.4.2. We conducted experiments with the same parameter settings as before, and the results are reported in Figure 12. We observe that the power consumption patterns of S5's Wi-Fi device is very similar to what we have observed in S3. By tuning the specific parameters for S5, we are now confident that PowerGuide can work equally well on the S5, as well as many other Wi-Fi chipsets based on 802.11n protocol.

VII. Conclusion

In this paper, we presented a FSM-based power estimation model for Wi-Fi communication on smartphones, and have implemented the model and its power estimation algorithm as an Android application called PowerGuide. Our experimental studies clearly showed the accuracy of PowerGuide compared to the popular power estimation tool PowerTutor. Furthermore, we have shown the appropriateness of our power model to the new generation of smartphones. As future work, we will tune PowerGuide to work with newest generation of smartphones and release PowerGuide to the research community. Moreover, we will conduct field tests with TCP flows and popular mobile applications.

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, Y. C. Hu, M. Zhang, P. Bahl, and Y.-M. Wang, "Fine-grained power modeling for smartphones using system call tracing," in *Proceedings of the sixth conference on Computer systems*, 2011, pp. 153–168.

[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] T. L. Cignetti, K. Komarov, and C. S. Ellis, "Energy estimation tools for the palm," in *Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, 2000, pp. 96–103.

[5] D. Kim, W. Jung, and H. Cha, "Runtime power estimation of mobile amoled displays," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE '13)*, 2013, pp. 61–64.

[6] C. Wang, F. Yan, Y. Guo, and X. Chen, "Power estimation for mobile applications with profile-driven battery traces," in *Low Power Electronics and Design (ISLPED), 2013 IEEE International Symposium on*, Sept 2013, pp. 120–125.

[7] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "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.

[8] 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.

[9] J. Lee, H. Joe, and H. Kim, "Automated power model generation method for smartphones," *Consumer Electronics, IEEE Transactions on*, vol. 60, no. 2, pp. 190–197, May 2014.

[10] L.-T. Duan, B. Guo, Y. Shen, Y. Wang, and W.-L. Zhang, "Energy analysis and prediction for applications on smartphones," *Journal of Systems Architecture*, vol. 59, no. 10, Part D, pp. 1375 – 1382, 2013.

[11] 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.

[12] 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.

[13] W. Jung, C. Kang, C. Yoon, D. Kim, and H. Cha, "Devscope: A nonintrusive and online power analysis tool for smartphone hardware components," in *Proceedings of the 8th International Conference on Hardware/Software Codesign and System Synthesis*, 2012, pp. 353–362.

[14] C. Yoon, D. Kim, W. Jung, C. Kang, and H. Cha, "Appscope: Application energy metering framework for android smartphones using kernel activity monitoring," in *Proceedings of the 2012 USENIX Conference on Annual Technical Conference (USENIX ATC '12)*, 2012, pp. 36–36.

[15] 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.

[16] L. Sun, R. K. Sheshadri, W. Zheng, and D. Koutsonikolas, "Modeling wifi active power/energy consumption in smartphones," in *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*, June 2014, pp. 41–51.

[17] PowerTutor. Accessed: 2015-01-07. [Online]. Available: http://powertutor.org/

[18] "BCM4334 Wi-Fi Chipset," https://www.broadcom.com/products/Wireless-LAN/802.11-Wireless-LAN-Solutions/BCM4334/, Broadcom, accessed: 2015-01-07.

[19] Monsoon Power Monitor. Accessed: 2015-01-07. [Online]. Available: http://msoon.com/

[20] Connectify Me. Accessed: 2015-01-07. [Online]. Available: http://www.connectify.me/

[21] SetCPU for Android. Accessed: 2015-01-07. [Online]. Available: http://www.setcpu.com/

[22] Y. Yang, C. Lim, and A. Nirmalathas, "Comparison of energy consumption of integrated optical-wireless access networks," in *Optical Fiber Communication Conference. Optical Society of America*, 2011.

[23] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, "Profiledroid: Multi-layer profiling of android applications," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking (MobiCom '12)*, 2012, pp. 137–148.

[24] "BCM4354 Wi-Fi Chipset," http://investor.broadcom.com/releasedetail.cfm?ReleaseID=827695, Broadcom, accessed: 2015-01-07.