Revisiting 802.11 Power Consumption Modeling in Smartphones

Swetank Kumar Saha, Pratham Malik, Selvaganesh Dharmeswaran, Dimitrios Koutsonikolas
University at Buffalo, The State University of New York
Buffalo, NY 14260-2500
Email: {swetankk, prathamm, selvagan, dimitrio}@buffalo.edu

***Abstract*—WiFi activity is a major source of power consumption in today's smartphones. Consequently, accurately WiFi power consumption models are extremely useful for researchers and app developers. Among a large number of models proposed recently, a model introduced by Serrano et al. was the first to add a new component – a per-frame energy toll incurred as a frame traverses the protocol stack – to the power consumption of the wireless NIC. The authors called this new component *cross-factor* and validated the accuracy of the model on a large number of devices, mostly 802.11g wireless routers and APs.**

This paper examines the validity of the model introduced by Serrano et al. on today's smartphones. We try to answer two questions: (i) Can the model accurately estimate the power consumption due to WiFi activity in today's smartphones given the complexity of modern smartphone architectures? (ii) Does the model remain valid in the case of 802.11n/ac interfaces, and if yes, can it reflect the impact of the new MAC features (e.g., MIMO, channel bonding) on the WiFi power consumption? Additionally, we study the impact of the power saving mode (PSM) which was ignored in the original model and show that ignoring PSM results in significant overestimation of the total power consumption at low frame generation rates. Accordingly, we propose a new model that works across the full range of frame generation rates and verify its accuracy for a wide range of parameters and devices.

I. Introduction

The proliferation of mobile devices (smartphones, tablets) has significantly increased the amount of data carried over wireless networks. Although the average cellular data rates have increased with the deployment of LTE, 802.11 WLANs still remain a primary choice for broadband Internet access due to their wide deployment and low or often zero cost to end users. With applications like Voice-over-IP (VoIP) gaining popularity and new proposals to offload LTE traffic to WiFi, the traffic carried over 802.11 networks is bound to grow.

Network connectivity through WiFi often incurs a major energy cost, which is a serious concern for battery-limited devices like smartphones. The problem is further exacerbated by the fact that the most popular smartphone applications need network access to function. It is hence not a surprise that a lot of research efforts have been targeted towards making 802.11 energy-efficient. Towards this end, 802.11 power consumption models are extremely useful for researchers/developers to help them and design power saving schemes or energy efficient applications. Over the past few years, many such power models have been proposed [1]-[2]. A common characteristic of most of them is that they treat the active (Tx or Rx) 802.11 power consumption as a sum of two components: a constant baseline one, plus a second one which reflects the power consumption of the 802.11 NIC.

The work by Serrano et al. [3] was the first to question this classical model, showing that there is an additional energy component incurred by the device itself when a packet traverses the protocol stack. This extra component, referred to as *cross-factor*, is independent of the packet size and the WiFi transmission parameters but accounts for a significant portion of the total power. The authors proposed a fine-grained power model for 802.11 devices that takes into account both the power consumption by the 802.11 NIC and by the protocol stack itself. Although the model was evaluated on a variety of WiFi devices, the focus was on routers/APs equipped with 802.11g interfaces. Only two 802.11g mobile devices were used and one 802.11n router.

Today, devices supporting the new IEEE 802.11n/ac standards have become much more common that those equipped with legacy WiFi interfaces (802.11a/b/g). These new standards offer much higher data rates by leveraging a MIMO-based PHY, coupled with Channel Bonding (CB), frame aggregation (FA), and more aggressive modulation and coding schemes (MCS). However, most of these high-throughput features also incur higher power consumption [4], [5], [6]. Additionally, today's smartphones have evolved and have a much more complex architecture, with more components bundled as part of a single System-On-Chip (SOC), compared to devices when the model was originally proposed. In the light of these changes, it becomes necessary to ask whether the original model can still act as an accurate power estimator across a range of today's mobile devices.

Additionally, battery-constrained mobile devices typically implement a Power Save mode (PSM); they remain at a very low power (sleep) state and they only switch to a high power state to transmit or receive packets. With low frame generation rates (common in smartphones where background processes typically contact their remote application servers to report status and get updates), it is expected that the NIC in such devices will remain in sleep state most of the time. Serrano et al. ignored PSM (a natural choice in the case of APs/routers, indeed the focus of [3]) and evaluated the model only for rates higher than 100 fps.

In this paper, we perform a detailed re-evaluation of the model from [3] to check its validity on today's smartphones. We experiment with a variety of 802.11g and 802.11n phones, in order to ensure that we are not profiling a single device. We make the following contributions:

- We show that the model accuracy remains very high across devices, network types (802.11g vs. 802.11n), and 802.11n configurations (combination of MCS, channel width, and number of spatial streams (NSS)).
- We investigate whether the trends w.r.t. the model parameters observed in [3] are still valid in the case of today's smartphones and whether they hold across devices and across 802.11 types. Additionally, in the case of 802.11n-equipped smartphones, we explore the impact of 802.11n MAC features (FA, CB, NSS) on the model parameters.
- We evaluate the original model for frame generation rates lower than 100 fps and show that the accuracy decreases significantly at very low rates, for both 802.11g and 802.11n devices. Additionally, we observe and report for first time, to our best knowledge, a non-standard sleep behavior of 802.11n NICS in a number of different smartphones; due to this behavior, the accuracy of the original model in the case of 802.11n is reduced even at relatively higher frame generation rates.
- Building upon the original model and our observations, we propose a new model that is applicable over the entire range of frame generation rates and across different network types.

II. Background

Power model The work in [3] modeled the total device power consumption due to WiFi transmissions as:

$$P = \rho_{id} + P_{tx} + P_{xg}(\lambda_g) = \rho_{id} + \rho_{tx}\tau_{tx} + \gamma_{xg}\lambda_g \quad (1)$$

- ρ_{id} is a platform-specific baseline power consumption. In [3], it is defined as "the power consumed in the "idle" state".
- $P_{tx} = \rho_{tx}\tau_{tx}$ is the power consumption of the 802.11 NIC which grows linearly with the transmission airtime percentage $\tau_{tx} = \lambda_g T_L$, where λ_g is the frame generation rate (kept below saturation) and $T_L = T_{PLCP} + (H+L)/R$ is the time required to transmit a frame of size L at a PHY bitrate R, accounting for the PLCP preamble T_{PLCP} and the MAC overhead H. [3] showed that ρ_{tx} increases with MCS, Tx power, and CPU frequency, in a large number of devices.
- $P_{xg}(\lambda_g) = \gamma_{xg}\lambda_g$ is the cross-factor, i.e., the power consumption incurred as the frames traverse the protocol stack, which depends only on the frame generation rate λ_g. The parameter γ_{xg} (in mJ/frame) is the per-frame processing energy toll as the frame crosses the protocol stack. [3] showed that γ_{xg} increases with CPU frequency but it is *independent* of the radio transmission parameters.

[3] developed a similar model for the total power consumption during WiFi reception. Additionally, the model was extended to account for retransmissions and ACKs. Since the Tx power consumption is higher than the Rx power consumption, especially in smartphones [3], [6], in this paper, we focus on the Tx power consumption. Further, we ignore ACKs and retransmissions, since [3] showed that they have a negligible impact on the radio power consumption and they do not affect the cross-factor.

PSM A WiFi radio in PSM switches between the sleep and the awake state. It remains in sleep state when there is no network activity and wakes up only at the beginning of every beacon period and any time it has a packet to transmit. In this paper, we focus on the most popular Dynamic PSM, which is used by all the devices in Table I. In Dynamic PSM, instead of entering the sleep state right after transmitting/receiving, the radio stays awake for a pre-defined duration called *PSM timeout* PSM_TO. The radio uses an *inactivity timer* and resets it to the PSM_TO value after every packet transmission/reception. The device notifies the AP of any transition to the awake or sleep state by sending a NULL data frame with the Power Management bit set to 0 or 1, respectively.

Clearly, for frame generation rates lower than a threshold $\lambda_{TH} = 1/PSM\_TO$, the packet inter-arrival time is longer than the PSM timeout. Thus, after each packet transmission, the radio will spend PSM_TO time idle and then it will switch to the sleep state. In contrast, when $\lambda_g \geq \lambda_{TH}$, the radio always remains in the awake state, as each packet transmission resets the inactivity timer before its expiration. Note that, based on the above description, we distinguish two sub-states in the awake state; *active* state (when the radio is transmitting or receiving) and *idle* state (the time between a packet transmission/reception and the expiration of the inactivity timer).

III. Experimental methodology

Our study was performed using five different smartphones (see Table I). We performed detailed power measurements with three of them – Nexus S, Galaxy S3, Galaxy S5. We used Nexus S for experiments with 802.11g, Galaxy S5 for experiments with 802.11n, and Galaxy S3 for experiments with both 802.11g and 802.11n. All these three phones are from one manufacturer: Samsung. In order to verify some important observations in Section V-B, we used two additional devices – G4 and Nexus 5X. First, to remove any H/W specific bias, we used a phone (G4) from a different manufacturer (LG). Finally, since the WiFi NICs in the Samsung and LG phones, although different, are all from the same chipset manufacturer (Broadcom Corporation), to further remove any S/W, driver, or WLAN card based biases, we also used another very recently released Nexus 5X phone that has a Qualcomm WiFi NIC.

We used a laptop running a Linux distribution (Ubuntu 12.04, kernel 3.6) as AP. The laptop was equipped with a Half Mini PCI-e Atheros AR9380 802.11a/b/g/n 3x3 WiFi adapter controlled by the open source ath9k [7] driver. All our experiments were done with Long Guard Interval (LGI); [8] found that SGI and LGI exhibit very little difference in terms of throughput and power consumption. To fix the MCS on the phones, we forced the AP to advertise support of a single MCS index in the 802.11n beacons.

TABLE I
SMARTPHONES USED IN OUR STUDY.

Manufacturer	Google [Samsung]	Samsung	Samsung	LG	Google [LG]
Model	Nexus S	Galaxy S3	Galaxy S5	G4	Nexus 5X
OS	Android 2.3.7 (CM 7.1)	Android 4.2.2 (CM 10.1.3)	Android 4.4.4 (CM 11.20)	Android 6.0	Android 6.0
Chipset	Broadcom BCM4329	Broadcom BCM4330	Broadcom BCM4354	Broadcom BCM4339	Qualcomm QCA6174
802.11g/802.11n	802.11g	Both	802.11n	802.11n [verify]	802.11n [verify]
802.11n features	N/A	40 MHz	40 MHz, MIMO 2x2	80 MHz, MIMO 2x2	80 MHz, MIMO 2x2

We measured power consumption on the phone using a Monsoon Power Monitor [9]. The power measurements were taken with the screen on, Bluetooth/GSM/3G radios disabled, and minimal background application activity, ensuring that the phone's base power is low and does not vary significantly over time. Each experiment involved a 10-second *iperf* session. The phone and the AP were placed very close to each other and we made sure there was no external interference.

CPU and Tx Power We wanted to minimize the impact of both the CPU frequency selection and Tx Power in our power measurements. Towards this end, we followed a simple strategy to select the appropriate CPU frequency. For each device and each WiFi configuration (802.11g/n, channel width, NSS), we started with the smallest CPU frequency offered by the device and then incremented it until we found the one that provides throughput at the maximum possible MCS (54 Mbps for 802.11g, MCS 7 for 802.11n/20SS and 40SS, MCS 15 for 802.11n/20DS and 40DS) comparable to what we could get with the highest CPU frequency setting available. This method guaranteed that the CPU does not become a bottleneck but at the same time it does not contribute any more to the phone's power consumption, than required to support the maximum possible throughput. Once CPU frequency is fixed, we used a similar strategy to select the lowest TX power setting that can support the highest throughput possible with a given configuration.

IV. MODEL APPLICABILITY IN SMARTPHONES

In this section, we study the applicability of the model in the case of 802.11g- and 802.11n-equipped smartphones. We only consider frame rates $\lambda_g \geq 100$ fps, similar to in [3].

A. Model construction and validation

Calculating the Tx airtime percentage The main challenge in building the model in the case of 802.11n is calculating the channel airtime percentage τ_{tx}. The time required to transmit a frame T_L can no longer be estimated as $T_L = T_{PLCP} + (H+L)/R$, because each transmitted frame may be of different frame size due to FA. Ignoring FA and assuming a constant frame size L can result in significant underestimation of τ_{tx}. On the other hand, using the maximum allowed AMPDU size may also result in large estimation errors; lower frame generation rates often yield lower AMPDU sizes, since typically WiFi drivers specify a maximum AMPDU size but they do not enforce it by delaying packets. Although one could capture the traffic on the receiver side and estimate the average aggregation size, we prefer a much simpler approach. In 802.11a/b/g, we have:

$$\tau_{tx} = \lambda_g T_L = \frac{\lambda_g}{1/T_L} = \frac{L\lambda_g}{L/T_L} \tag{2}$$

where $Th = L\lambda_g$ is the application layer throughput and L/T_L is the MAC layer data rate which is also equal to the maximum possible application layer throughput ($\lambda_g \leq 1/T_L$). Hence, we can approximate the airtime percentage as

$$\tau_{tx} = \frac{Th}{Th_{max}} \tag{3}$$

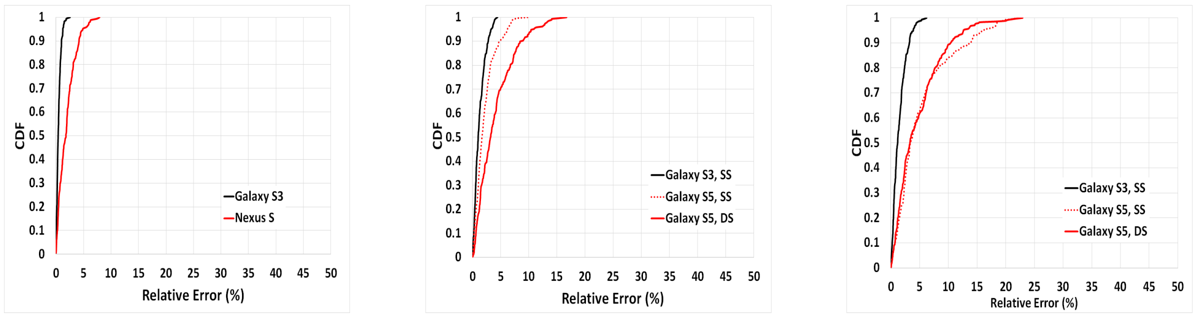
where Th_{max} is the maximum possible application layer throughput which we measure by saturating the link with UDP traffic of maximum packet size.

Measurements For 802.11g devices, we considered 3 different PHY bitrates (12 Mbps, 24 Mbps, and 48 Mbps). For each bitrate, we took measurements of the total device power consumption for combinations of 8 frame sizes $L \in [100, 2000]$ bytes and 7 frames rates $\lambda_g \in [100, 1800]$ fps. For 802.11n devices, we built the model separately for each supported channel width and NSS. For each combination (20SS, 40SS, 20DS, 40DS), we considered 3 different MCS (1, 4, and 6 in the case of SS and 9, 12, and 14 in the case of DS). For each MCS, we measured the total device power consumption for the same 8 frame sizes as in the case of 802.11g and 15 frame generation rates $\lambda_g \in [100, \lambda_g^{max}]$ fps, where λ_g^{max} is the maximum supported frame rate, different for each (channel width, NSS, MCS) combination. To build the model, we used simple linear regression following the methodology of [3].

Evaluation Figure 1 shows the cumulative distribution functions (CDFs) of the model's relative errors with each phone in 802.11g and 802.11n. The relative error is defined as $E_{rel} = \frac{|P_{model} - P_{measure}|}{P_{measure}}$, where $P_{measure}$ is the power consumption measured with Monsoon and P_{model} is the estimated value from equation (1) using the parameters from Table II. We observe that the errors remain very low for all three phones and all WiFi configurations. In particular, in the case of 802.11g, the errors remain below 3% with Nexus S and below 8% with Galaxy S3. In the case of 802.11n, the errors with S3 increase only sightly. The errors with Galaxy S5 are higher – the maximum error is lower than 17% (23%) at 20 MHz (40 MHz) – but the 90th percentile still remains below 14% with all 4 configurations. *Overall, we conclude that the model remains valid for modern smartphones in both 802.11g and 802.11n. Interestingly though, the error tends to increase for configurations that support higher throughputs. This raises a concern for 802.11ac that supports even wider channels.*

B. Model parameters

We now take a closer look at the model parameters and study their properties and their impact on the total power consumption. Table II lists the values of the model parameters for each device.



(a) 802.11g. (b) 802.11n (channel width 20 MHz). (c) 802.11n (channel width 40 MHz).

Fig. 1. CDFs of the relative error with Nexus S, Galaxy S3, and Galaxy S5 for $\lambda_g \geq 100$ fps.

TABLE II
POWER MODEL PARAMETERS FOR ALL THE DEVICES UNDER STUDY.

	$\rho_{id}(mW)$	$\rho_{tx}(mW)$	$\gamma_{xg}(mJ)$
	Nexus S (802.11g)		
$12M$	$794.79 \pm 0.93\%$	$753.64 \pm 3.45\%$	$0.14 \pm 5.64\%$
$24M$	$792.97 \pm 0.94\%$	$796.03 \pm 3.68\%$	$0.12 \pm 8.24\%$
$48M$	$794.55 \pm 0.94\%$	$1009.74 \pm 4.83\%$	$0.10 \pm 11.35\%$
	Galaxy S3 (802.11g)		
$12M$	$934.22 \pm 0.21\%$	$750.84 \pm 0.96\%$	$0.05 \pm 4\%$
$24M$	$936.25 \pm 0.15\%$	$746.72 \pm 0.78\%$	$0.04 \pm 5.28\%$
$48M$	$955.86 \pm 0.21\%$	$734.63 \pm 1.95\%$	$0.06 \pm 4.91\%$
	Galaxy S3 (802.11n, 20 MHz, SS)		
$MCS1$	$968.71 \pm 0.31\%$	$606.13 \pm 1.38\%$	$0.12 \pm 3.37\%$
$MCS4$	$977.52 \pm 0.34\%$	$558.45 \pm 1.42\%$	$0.09 \pm 2.32\%$
$MCS6$	$973.38 \pm 1.7\%$	$493.49 \pm 9.87\%$	$0.08 \pm 11.96\%$
	Galaxy S3 (802.11n, 40 MHz, SS)		
$MCS1$	$1233.16 \pm 0.19\%$	$613.51 \pm 1.14\%$	$0.14 \pm 2.15\%$
$MCS4$	$1258.36 \pm 0.34\%$	$465.19 \pm 3.2\%$	$0.11 \pm 1.84\%$
$MCS6$	$1276.97 \pm 0.49\%$	$370.6 \pm 6.68\%$	$0.09 \pm 2.12\%$
	Galaxy S5 (802.11n, 20 MHz, SS)		
$MCS1$	$1053.91 \pm 0.87\%$	$1208.34 \pm 1.78\%$	$0.22 \pm 4.95\%$
$MCS4$	$1063.1 \pm 0.62\%$	$816.26 \pm 1.88\%$	$0.12 \pm 3.23\%$
$MCS6$	$1088.8 \pm 0.8\%$	$750.56 \pm 3.57\%$	$0.10 \pm 4.03\%$
	Galaxy S5 (802.11n, 40 MHz, SS)		
$MCS1$	$1140.83 \pm 1.17\%$	$872.22 \pm 4.06\%$	$0.14 \pm 9.77\%$
$MCS4$	$1275.51 \pm 1.39\%$	$797.25 \pm 8.07\%$	$0.06 \pm 11.17\%$
$MCS6$	$1310.96 \pm 1.34\%$	$704.55 \pm 10.05\%$	$0.04 \pm 11.45\%$
	Galaxy S5 (802.11n, 20 MHz, DS)		
$MCS9$	$1055.97 \pm 0.69\%$	$830.1 \pm 2.27\%$	$0.17 \pm 4.82\%$
$MCS12$	$1153.65 \pm 1.02\%$	$597.31 \pm 6.82\%$	$0.07 \pm 7.65\%$
$MCS14$	$1153.7 \pm 1.08\%$	$625.73 \pm 8.08\%$	$0.06 \pm 6.84\%$
	Galaxy S5 (802.11n, 40 MHz, DS)		
$MCS9$	$1225.91 \pm 0.61\%$	$665.31 \pm 3.32\%$	$0.13 \pm 3.1\%$
$MCS12$	$1327.49 \pm 1.07\%$	$647.9 \pm 10.21\%$	$0.06 \pm 6.31\%$
$MCS14$	$1394.82 \pm 1.23\%$	$692.89 \pm 11.4\%$	$0.04 \pm 9.11\%$

Idle power Table II shows that the idle power is different across different devices and across different 802.11 modes for the same device (higher in 802.11n than in 802.11g). Additionally, for devices in 802.11n mode, the idle power with a 40 MHz channel is higher than with a 20 MHz channel for a given MCS and NSS (27-31% in Galaxy S3 and 8-21% in Galaxy S5); this finding was also reported in [5], [6]. On the other hand, activating a 2nd spatial stream has a minimal impact on the idle power consumption (at most 8%); this rather counter-intuitive result was also reported in [6].

Properties of γ_{xg} One of the main findings in [3] was that γ_{xg} is constant for a given device (and a fixed CPU frequency), independent of the WiFi transmission parameters. Table II shows that this is no the case for the devices we study in this paper. For a given device, γ_{xg} differs among different MCS by 2 mJ/frame (Galaxy S3, 802.11g) up to 12 mJ/frame (Galaxy S5, 802.11n/20SS). Further, γ_{xg} is higher in 802.11n than in 802.11g for the same device (Galaxy S3), and for 802.11n devices, it takes different values across different combinations of channel width and NSS. We also observe that there is no clear trend across devices. In Galaxy S3, γ_{xg} is slightly higher at 40SS than at 20SS. In contrast, in Galaxy S5, it takes the highest values at 20SS and the lowest ones at 40DS.

In spite of the discrepancy in the γ_{xg} values for the same device, we note that these values are order of magnitudes smaller than the values of ρ_{tx} and hence, their impact on the total power consumption may be limited. If that is the case, then one could still use one γ_{xg} value for all MCS (e.g., the average γ_{xg} across all MCS) without affecting the accuracy. To investigate this, Table III shows the average relative error of the power model for each MCS when we replace in the model the γ_{xg} value of that MCS with the γ_{xg} values of other MCS as well as with the average γ_{xg} value across the 3 MCS we consider in Table II. The diagonal cells (in bold font) show the errors with the correct γ_{xg} values.

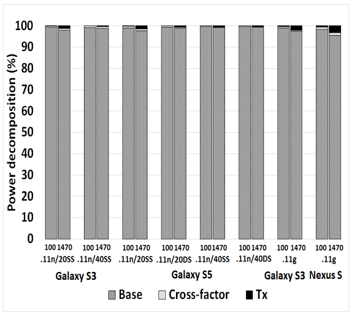
Table III shows that using a different γ_{xg} in 802.11g does not affect the accuracy; the maximum value for the average relative error is only 3.9% (Nexus S, 48 Mbps with $\gamma_{xg_{12M}}$). In fact, the error with $\gamma_{xg_{avg}}$ is always less than 3%. Hence, we can use $\gamma_{xg_{avg}}$ in equation (1) and be consistent with [3], i.e., have a single γ_{xg} independent of the MCS. On the other hand, in 802.11n, the average relative error can sometimes be as high as 20% with standard deviations of similar levels (e.g., Galaxy S5, 40DS, MCS 14 with γ_{xg_9}). However, the errors with $\gamma_{xg_{avg}}$ never exceed 11% (with a standard deviation of up to 9.55%) and, in many cases, they remain lower than 5%. Hence, depending on the desired level of accuracy, we can still use $\gamma_{xg_{avg}}$ in equation (1).

Properties of ρ_{tx} The authors in [3] found that for a large number of 802.11g devices including routers, APs, tablets, and smartphones, and one 802.11n wireless router, ρ_{tx} increased with the MCS. In contrast, Table II shows that the relationship between ρ_{tx} and MCS is very different for different devices. Among all the devices in Table II, only the oldest Nexus S phone exhibits the same trend with ρ_{tx} increasing from 753 mW to 1009 mW as the bitrate increases from 12 Mbps to 48 Mbps. On the other hand, in Galaxy S3 (both in 802.11g and 802.11n mode) and in Galaxy S5 (at 20SS and 40SS), ρ_{tx} *decreases* with the MCS. The same device (S5) exhibits a different trend in the case of DS (both 20DS and 40DS) with both MCS 9 and 14 having a higher ρ_{tx} than MCS 12.

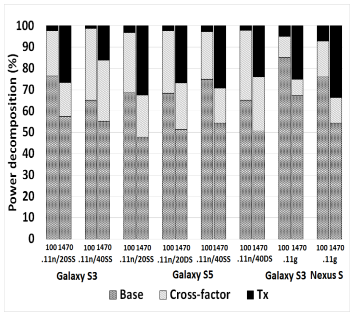
TABLE III
SUMMARY OF RELATIVE ERRORS (AVERAGE AND STANDARD DEVIATION) WITH DIFFERENT γ_{xg} VALUES.

802.11g smartphones					
	$BitRate$	$\gamma_{xg_{12M}}$	$\gamma_{xg_{24M}}$	$\gamma_{xg_{48M}}$	$\gamma_{xg_{avg}}$
Nexus S	$\mathbf{12M}$	$\mathbf{1.69\% \pm 1.34\%}$	$1.96\% \pm 1.79\%$	$3.17\% \pm 2.73\%$	$2.02\% \pm 1.83\%$
	$\mathbf{24M}$	$2.53\% \pm 2.14\%$	$\mathbf{1.99\% \pm 1.74\%}$	$2.36\% \pm 2.18\%$	$2\% \pm 1.73\%$
	$\mathbf{48M}$	$3.9\% \pm 3.41\%$	$2.8\% \pm 2.48\%$	$\mathbf{2.23\% \pm 1.75\%}$	$2.75\% \pm 2.42\%$
Galaxy S3	$\mathbf{12M}$	$\mathbf{0.47\% \pm 0.4\%}$	$0.95\% \pm 0.73\%$	$0.83\% \pm 0.69\%$	$0.48\% \pm 0.4\%$
	$\mathbf{24M}$	$0.94\% \pm 0.67\%$	$\mathbf{0.39\% \pm 0.31\%}$	$1.72\% \pm 1.14\%$	$0.92\% \pm 0.65\%$
	$\mathbf{48M}$	$0.99\% \pm 0.78\%$	$1.79\% \pm 1.31\%$	$\mathbf{0.59\% \pm 0.5\%}$	$1.01\% \pm 0.79\%$

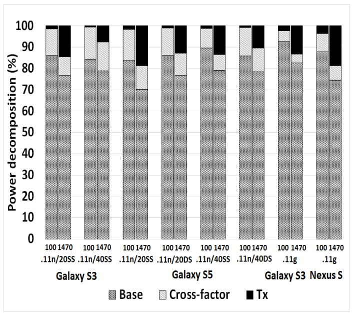
802.11n smartphones					
	MCS	γ_{xg_1} / γ_{xg_9}	γ_{xg_4} / $\gamma_{xg_{12}}$	γ_{xg_6} / $\gamma_{xg_{14}}$	$\gamma_{xg_{avg}}$
Galaxy S3 20 MHz, SS	**1**	$\mathbf{0.71\% \pm 0.51\%}$	$2.2\% \pm 1.47\%$	$2.86\% \pm 1.9\%$	$1.77\% \pm 1.22\%$
	4	$4.17\% \pm 2.38\%$	$\mathbf{1.05\% \pm 0.71\%}$	$1.77\% \pm 1.08\%$	$1.27\% \pm 0.97\%$
	6	$5.8\% \pm 4.79\%$	$1.74\% \pm 1.89\%$	$\mathbf{1.73\% \pm 1.2\%}$	$2.27\% \pm 2.62\%$
Galaxy S3 40 MHz, SS	**1**	$\mathbf{0.5\% \pm 0.37\%}$	$1.92\% \pm 1.33\%$	$2.77\% \pm 1.87\%$	$1.6\% \pm 1.13\%$
	4	$4.03\% \pm 2.72\%$	$\mathbf{1.34\% \pm 0.9\%}$	$2.49\% \pm 1.33\%$	$1.39\% \pm 1.08\%$
	6	$6.94\% \pm 5.01\%$	$2.59\% \pm 2.37\%$	$\mathbf{2.09\% \pm 1.37\%}$	$3.25\% \pm 2.8\%$
Galaxy S5 20 MHz, SS	**1**	$\mathbf{1.76\% \pm 1.33\%}$	$5.15\% \pm 3.06\%$	$6.18\% \pm 3.8\%$	$4.13\% \pm 2.38\%$
	4	$10.42\% \pm 5.42\%$	$\mathbf{1.7\% \pm 1.55\%}$	$3.46\% \pm 1.61\%$	$2.96\% \pm 2.06\%$
	6	$16.21\% \pm 9.49\%$	$3.87\% \pm 3.08\%$	$\mathbf{2.82\% \pm 2.1\%}$	$6.81\% \pm 4.27\%$
Galaxy S5 40 MHz, SS	**1**	$\mathbf{2.68\% \pm 2.2\%}$	$5.79\% \pm 3.54\%$	$6.78\% \pm 4.22\%$	$4.79\% \pm 2.85\%$
	4	$13.47\% \pm 8.39\%$	$\mathbf{5.61\% \pm 5\%}$	$6.85\% \pm 4.77\%$	$6.48\% \pm 5.06\%$
	6	$20.79\% \pm 15.35\%$	$7.55\% \pm 5.69\%$	$\mathbf{6.39\% \pm 5.39\%}$	$10.58\% \pm 6.97\%$
Galaxy S5 20 MHz, DS	**9**	$\mathbf{1.67\% \pm 1.25\%}$	$6.81\% \pm 3.89\%$	$7.21\% \pm 4.16\%$	$4.95\% \pm 2.71\%$
	12	$16.55\% \pm 11.17\%$	$\mathbf{4.4\% \pm 3.37\%}$	$4.66\% \pm 3.39\%$	$6.49\% \pm 4.67\%$
	14	$21.86\% \pm 16.37\%$	$5.25\% \pm 3.57\%$	$\mathbf{5.08\% \pm 3.57\%}$	$8.6\% \pm 7.04\%$
Galaxy S5 40 MHz, DS	**9**	$\mathbf{2.09\% \pm 1.63\%}$	$8.04\% \pm 3.7\%$	$10.1\% \pm 4.88\%$	$6.41\% \pm 2.86\%$
	12	$13.96\% \pm 11.42\%$	$\mathbf{5.28\% \pm 3.63\%}$	$7.44\% \pm 3.93\%$	$5.87\% \pm 4.47\%$
	14	$22.51\% \pm 19.68\%$	$8.08\% \pm 6.57\%$	$\mathbf{6.63\% \pm 5.11\%}$	$10.52\% \pm 9.55\%$



(a) $\lambda_g = 100$ fps.



(b) $\lambda_g = \lambda_g^{max}$.



(c) Average power decomposition.

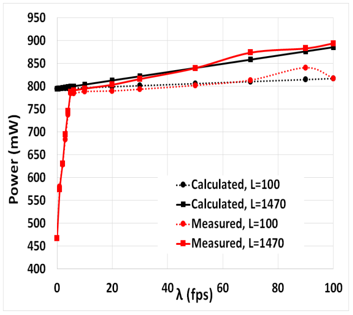
Fig. 2. Total power decomposition for different devices, WiFi configurations, and packet sizes. The PHY bitrate is set to 48 Mbps in 802.11g and the MCS is set to 6/14 in 802.11n SS/DS.

In the case of 802.11n devices, we are also interested in the relationship between ρ_{tx} and the other two WiFi transmission parameters, channel width and NSS for a given MCS. As far as the channel width is concerned, Table II shows that ρ_{tx} is lower at 40 MHz than at 20 MHz in both devices (S3 and S5) with a SS. On the other hand, in the case of Galaxy S5 with DS, it decreases substantially (from 830 mW to 665 mW) only for MCS 9 but increases sightly for MCS 12 and 14. As far as the NSS is concerned, ρ_{tx} decreases when we move from SS to DS for any MCS and channel width.

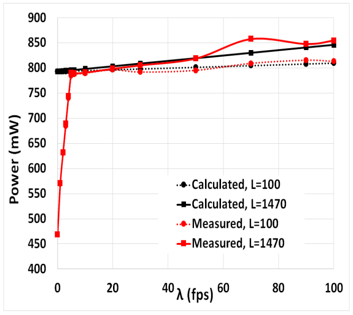
Summarizing, a general trend appears in Table II: In modern smartphones, higher throughput configurations (higher MCS, wider channels, higher NSS) appear to have (with a few exceptions) lower ρ_{tx}. Consequently, contrary to the common belief, such configurations contribute less to the WiFi power consumption and to the overall device power consumption. However, remember that 40 MHz channel can contribute a significant amount to the idle power consumption, which often dominates the total power consumption (Figure 2). The same conclusion was reported in a measurement study in [5] for 802.11ac. This result shows the need for new power saving schemes that can reduce the power consumption of the idle mode operation in wider channels.

Power consumption breakdown In order to gain insights on the contribution of each of the three power components on the total power consumption in different devices, we examine in Figure 2 how the total power consumption in each device is decomposed into three parts: the base (idle) power ρ_{id}, the cross-factor $P_{xg}(\lambda_g) = \gamma_{xg}\lambda_g$, and the WiFi transmission power $P_{tx} = \rho_{tx}\tau_{tx}$. We show the results for 802.11g with a PHY bitrate of 48 Mbps and 802.11n with MCS 6/14, and two different packet sizes: 100 bytes and 1470 bytes. To understand the impact of λ_g on the cross-factor and the WiFi transmission power, we plot the result for $\lambda_g = 100$ fps in Figure 2(a), $\lambda_g = \lambda_g^{max}$ (different for each WiFi configuration) in Figure 2(b), and the average result over all $\lambda_g \in [100, \lambda_g^{max}]$ in Figure 2(c).

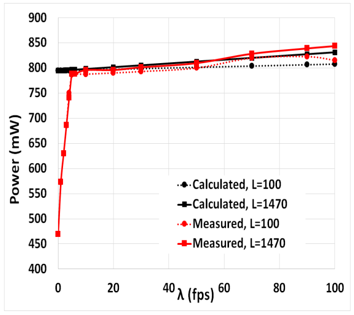
A first observation from Figures 2(a), 2(b), 2(c) is that the idle/base power is the largest component of the total power consumption. On average (Figure 2(c)), p_{id} contributes 80-93% of the total power in the case of 100-byte packets and 70-82% in the case of 1470-bytes packets. More importantly,



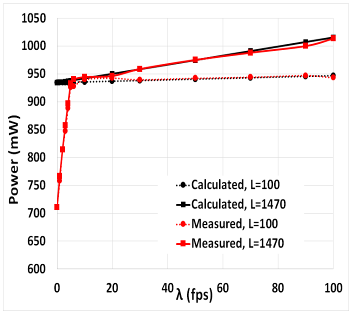
(a) Nexus S (12 Mbps).



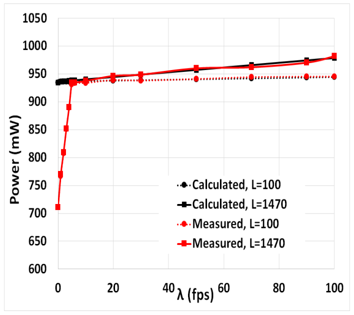
(b) Nexus S (24 Mbps).



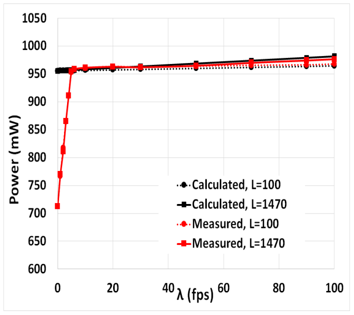
(c) Nexus S (48 Mbps).



(d) Galaxy S3 (12 Mbps).



(e) Galaxy S3 (24 Mbps).



(f) Galaxy S3 (48 Mbps).

Fig. 3. Comparison of measured and calculated power in Nexus S and Galaxy S3 (802.11g) for $\lambda_g \in [0, 100]$.

in the case of low frame generation rates (Figure 2(a)) the contribution of p_{id} increases to 96-99%, making the contribution of both the cross-factor and the WiFi transmission power negligible. On the other hand, in the case of high frame generation rates (Figure 2(b)), the weight of p_{id} is lower, although it still constitutes 50-85% of the total power. If we ignore p_{id} for a moment and focus on the relative contribution of the other two components, we observe that the cross-factor dominates with small packet sizes and the WiFi Tx component dominates with large packet sizes. The same observation was made in [3] for a smartphone and a tablet device. Specifically, in the average case, the cross-factor contributes 7-15% of the total power consumption with 100-byte packets and 5-12% with 1470-byte packets. In the case of high λ_g, its contribution increases up to 33% in 802.11n but remains low in 802.11g.

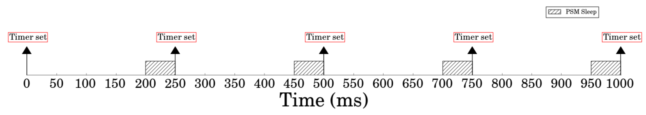
V. Impact of PSM

In the last section, we limited our study to frame generation rates $\lambda_g \geq 100$ fps. In practice, low frame generation rates are possible and generally occur when the smartphone screen is off but background processes are actively communicating with their remote servers for regular reporting of status and fetching of updates. Since such background processes can account for a large portion of the total battery drain in a day [**?**], it is important to understand the power consumption in the low activity region. Hence, in this section, we take a look at the applicability of the original model for $\lambda_g < 100$ fps, in both 802.11g and 802.11n.

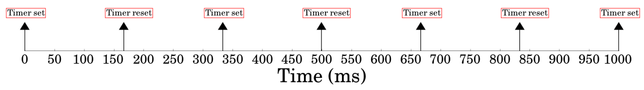
A. 802.11g

Figure 3 plots the measured and calculated (using the original model (1)) power for Nexus S and Galaxy S3 respectively, as a function of $\lambda_g \in [0, 100]$ fps, for three PHY bitrates and two packet sizes (100B and 1470B). We observe that, for each of the three bitrates and any of the two packet sizes (for both Nexus S and Galaxy S3), the model has very minimal or zero deviation from the measured power for $5 < \lambda_g \leq 100$ fps. In contrast, the measured power below the 5 fps threshold drops much more sharply as we lower the frame generation rate. As a result, the original model overestimates significantly the power consumption for $\lambda_g \in [0, 5)$ and the error increases as λ_g decreases. In the worst case, the error can be as high as 330 mW for Nexus S and 240 mW for Galaxy S3. Note that, although we only show plots for two packet sizes, we have confirmed that other packet sizes exhibit the same trend, only differing marginally in their absolute values.

1) Understanding power consumption for $\lambda_g < 5$ fps: By looking at packet traces collected from all the experiments involving frame generation rates $\lambda_g < 5$ fps, we observed that both devices use the 802.11 Dynamic PSM (see Section II). As we described in Section II, the radio switches to the sleep state when the packet inter-arrival time is longer than the PSM timeout PSM_TO. 802.11 does not specify the value of PSM_TO. By analyzing packet traces, we empirically found that, for both 802.11g devices used in our experiments, PSM_TO is set to *200 ms*, which corresponds to a frame generation rate $\lambda_{TH} = 5$ fps. Hence, for $\lambda_g < \lambda_{TH} = 5$ fps, the radio spends an amount of time at the low power sleep state instead of the high power idle state; this behavior is not captured by the original model, resulting in low accuracy. In contrast, for $\lambda_g \geq \lambda_{TH}$, a new packet is always scheduled for transmission before the completion of the PSM timeout, resetting the inactivity timer (indeed, we found no NULL frames with the power management bit set to 1 in our traces for $\lambda_g \geq 5$ fps). As a result, the radio never switches to the



(a) The radio switches to the sleep state for $\lambda_g = 4$ fps.



(b) The radio remains always awake for $\lambda_g = 6$ fps.

Fig. 4. Illustration: PSM in 802.11g for $\lambda_g < \lambda_{TH}$ and $\lambda_g \geq \lambda_{TH}$ fps.

sleep state and the original model can estimate correctly the power consumption.

Figure 4 illustrates a typical sleep/transmission time-line for $\lambda_g = 4 < 5$ fps (4(a)) and $\lambda_g = 6 > 5$ fps (4(b)). The arrows represent packet transmissions from the phone to the AP. For $\lambda_g = 4$ fps, the inter-arrival time between the two consecutive packets is 250 ms, greater than the PSM timeout of 200 ms. Thus, the device goes to sleep after 200 ms and wakes up when the next packet is ready for transmission (after 50 ms in this example). Contrast this to the case with $\lambda_g = 6$ fps, where the packet inter-arrival time is reduced to 166.66 ms. This causes the timer to be reset before its expiration and the device never enters the sleep state.

2) Corrected power model: We now modify the original power model P_{orig} (equation (1)) to provide a corrected model $P_{corrected}$, which takes into account PSM behavior. To account for the reduced power consumption cased due to sleeping, we first calculate the number of sleep intervals n_s in a given time interval T as

$$n_s = \lfloor T \cdot \lambda_g \rfloor \tag{4}$$

and the amount of time T_s the interface sleeps in each such interval as

$$T_s = \frac{1}{\lambda_g} - \frac{1}{\lambda_{TH}} \tag{5}$$

Given the sleep power P_{sleep}, obtained experimentally for each device, we model the power consumption as:

$$\begin{aligned} P_{corrected} &= \frac{P_{orig}(T - n_s \cdot T_s) + P_{sleep}(n_s \cdot T_s)}{T} \\ &= P_{orig} \cdot \frac{\lambda_g}{\lambda_{TH}} + P_{sleep} \cdot (1 - \frac{\lambda_g}{\lambda_{TH}}) \end{aligned} \tag{6}$$

Equation (6) is valid only when PSM comes into play, for $\lambda_g < \lambda_{TH}$. In contrast, when $\lambda_g \geq \lambda_{TH}$, the original model (1) does not require any correction. The two equations can be combined into one that works with the entire range of frame generation rates as

$$P_{corrected} = P_{orig} \cdot min(1, \frac{\lambda_g}{\lambda_{TH}}) + P_{sleep} \cdot (1 - min(1, \frac{\lambda_g}{\lambda_{TH}})) \tag{7}$$

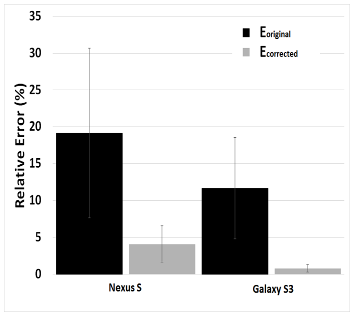
3) Accuracy of $P_{corrected}$ (802.11g): Figure 5 compares the average relative error of the original model (1) and the corrected model (7), $E_{original}$ and $E_{corrected}$, respectively, with each of the two 802.11g phones for three PHY bitrates and $0 \leq \lambda_g < 5$ fps (the two models are identical for $\lambda_g \geq 5$ fps). The average is taken over many different λ_g values and two different packet sizes (100B and 1470B). We observe that the original model fails to accurately estimate the power consumption in the low λ_g region with average errors as high as 19% for Nexus S and 13% for Galaxy S3. In contrast, our corrected model accounting for the time spent in the sleep state achieves very high accuracy, keep the average error lower than 4% for Nexus S and lower than 1.5% for Galaxy S3.

B. 802.11n

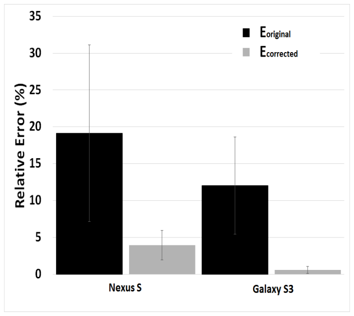
Figure 6 plots the measured and calculated (using the original model) power as a function of $\lambda_g \in [0, 100]$ fps for Galaxy S3 (20SS, MCS 1, 4 and 6) and Galaxy S5 (40DS, MCS 9, 12 and 14), for two packet sizes 100B and 1470B. The behavior for Galaxy S3 (40SS) and Galaxy S5 (20SS, 40SS, 20DS) is similar – we omit these figures in the interest of space. For both phones and all MCS, packet size, and channel width combinations, we observe a behavior similar to that of 802.11g for $0 \leq \lambda_g < 5$ fps. However, different from 802.11g, in Figure 6 we observe significant errors even for $\lambda_g > 5$ fps all the way up to $\lambda_g < 100$ fps. Interestingly, for $5 \leq \lambda_g < 30$ fps, the measured power varies in a *non-linear, zig-zag* fashion, increasing with λ_g for some regions and then decreasing at certain frame generation rates. This zig-zag pattern diminishes for higher λ_g but the disparity between the calculated (from the model) and measured values remains even for the region of $30 \leq \lambda_g < 100$ fps (in contrast, in Figure 3 we saw that the error becomes negligible for $\lambda \geq 5$ fps).

1) Understanding power consumption for $\lambda_g < 100$ fps: To get a deeper insight into this behavior specific to 802.11n devices, we looked again at their corresponding packet traces. We observed again the devices spending time in the PSM sleep state, similar to the 802.11g case. In fact, the PSM timeout for Galaxy S5 is also equal to 200 ms. However, *we further observed and report for first time to our knowledge, that both 802.11n devices follow a different sleep strategy compared to the standard strategy used by 802.11g devices* (in particular, Galaxy S3 implements different strategies in 802.11g and 802.11n mode). Specifically, the inactivity timer is not reset after a packet transmission and the radio always enters the sleep state after the timer expires regardless of the number of packet transmissions during the PSM timeout interval (and hence, regardless of the frame generation rate). In order to make sure we were not observing an energy bug or H/W platform specific issues, we confirmed this behavior with two more devices: LG G4 and Nexus 5X. Both LG G4 and Nexus 5X indeed showed similar behavior and similar patterns for the measured power for $0 \leq \lambda_g \leq 100$ fps (although Nexus S uses a much more aggressive PSM timeout of only 8 ms).

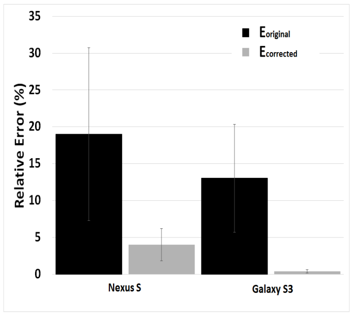
Figure 7 shows an example of the sleep behavior as observed for a 802.11n device for $\lambda_g = 6$ fps. The device sends a packet at $t = 0$ ms and starts its inactivity timer ($PSM\_TO = 200$ ms). At $t = 166.66$ ms, another uplink packet is scheduled but it does not reset the timer. The radio enters the sleep state at $t = 200$ ms, when the inactivity timer set after the first packet transmission (t=0) expires. Compare this to 802.11g (standard) behavior for the same



(a) 12 Mbps

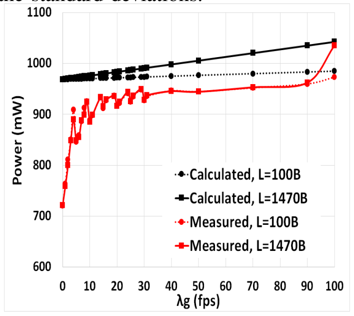


(b) 24 Mbps

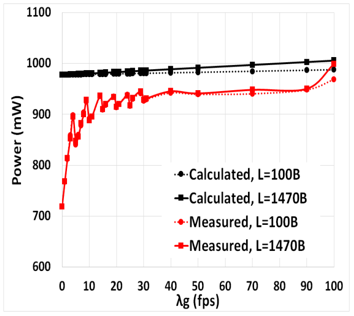


(c) 48 Mbps

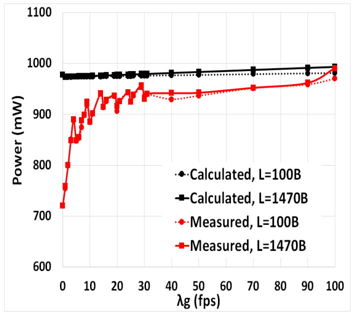
Fig. 5. Average relative error of the original and the corrected model for Nexus S and Galaxy S3 (802.11g) and three different bitrates. The error bars represent the standard deviations.



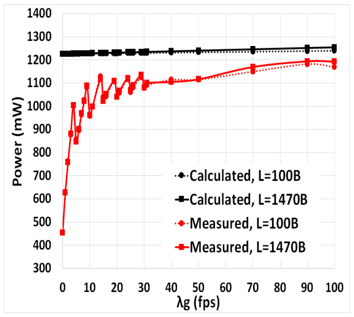
(a) Galaxy S3 (20SS, MCS1).



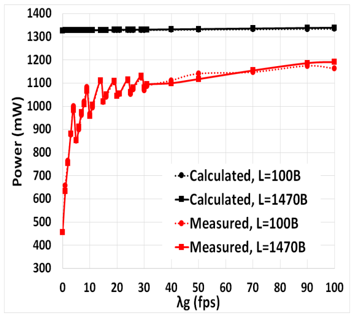
(b) Galaxy S3 (20SS, MCS4).



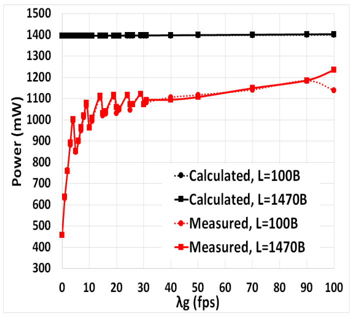
(c) Galaxy S3 (20SS, MCS6).



(d) Galaxy S5 (40DS, MCS9).



(e) Galaxy S5 (40DS, MCS12).



(f) Galaxy S5 (40DS, MCS14).

Fig. 6. Comparison of measured and calculated power in Galaxy S3 (20SS) and Galaxy S5 (40DS) for $\lambda_g \in [0, 100]$.

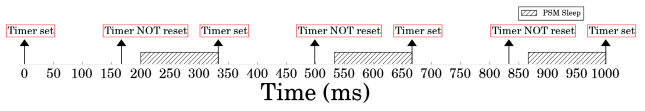


Fig. 7. Illustration: PSM in 802.11n for $\lambda_g = 6$ fps. Packet transmissions do not cancel the inactivity timer. The radio goes to sleep although $\lambda_g > \lambda_{TH}$.

frame generation rate (Figure 4(b)), where the packet at $t = 166.66$ ms resets the inactivity timer, preventing the radio from entering the sleep state.

2) *Adapting the corrected model for 802.11n:* Based on the observations described above, we adapt the power model $P_{corrected}$ introduced in Section V-A2. Using the same notations, we update the definition of the number of sleep intervals in a given time interval T to:

$$n_s = \lfloor \frac{T \cdot \lambda_g}{k} \rfloor \tag{8}$$

where k is a positive integer defined as $k = \lfloor \frac{\lambda_g}{\lambda_{TH}} + 1 \rfloor$.

The amount of time T_s the interface sleeps in each sleep interval is now calculated as

$$T_s = \frac{k}{\lambda_g} - \frac{1}{\lambda_{TH}} \tag{9}$$

Finally, we model the power consumption as

$$\begin{aligned} P_{corrected} &= \frac{P_{orig}(T - n_s \cdot T_s) + P_{sleep}(n_s \cdot T_s)}{T} \\ &= P_{orig} \cdot \frac{\lambda_g}{k \cdot \lambda_{TH}} + P_{sleep} \cdot (1 - \frac{\lambda_g}{k \cdot \lambda_{TH}}) \end{aligned} \tag{10}$$

Equation (10) holds for *any* λ_g for 802.11n devices implementing the PSM behavior described in V-B1. Note that for $\lambda_g < \lambda_{TH}$, (8) gives $k = 1$ and equation (10) becomes identical to (7), i.e., the new model defaults to the 802.11g model which describes the standard PSM behavior.

Figure 8 plots the number of sleep intervals and the duration of the sleep interval (equations (8) and (9), respectively) as a function of λ, for $T = 10$ sec (the duration of each experiment). We observe that both n_s and T_s exhibit a zig-zag pattern which explains the zig-zag pattern of the measured

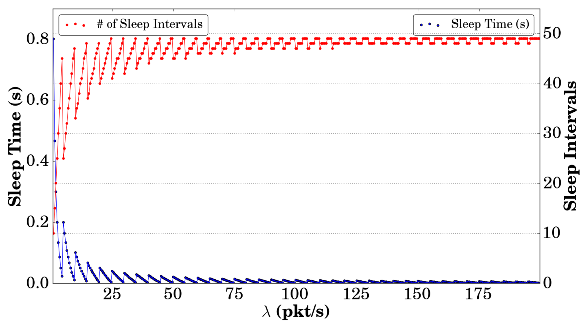


Fig. 8. Duration of sleep interval T_s and number of sleep intervals n_s during a 10-sec period as a function of λ_g. We assume $\lambda_{TH} = 5$ fps.

power consumption in Figures 6(a)-6(f). We also observe that, for $\lambda_g \geq 100$ fps, T_s becomes very short while n_s remains close to 50. Hence, the total time the NIC spends in the sleep state approaches zero and the impact of PSM on the total power consumption becomes negligible (In equation (10) $P_{corrected} \approx P_{orig}$ when $T_s \approx 0$).

3) Accuracy of $P_{corrected}(802.11n)$: Figure 9 compares the average relative error of the original and the corrected model (equation (10)) with Galaxy S3 and S5 for three different MCS, all combinations of channel width and NSS, and $0 \leq \lambda_g < 100$ fps. We observe that the average error with the original model is even higher compared to the 802.11g case, especially with Galaxy S5, often exceeding 60%, with large standard deviations. Although the corrected model also exhibits higher errors compared to the 802.11 case, it remains much more accurate than the original model for all WiFi configurations; its average error never exceeds 18% and the standard deviations always remain low. Note that, in contrast to 802.11g, here the two models exhibit large disparity in their accuracy even for $\lambda_g \geq 5$ fps, since the radio always enters the sleep state. Also, note that the error of the corrected model remains constant over the whole range of $\lambda_g \in [0, 100]$ for a given WiFi configuration. In contrast, the error of the original model drops with λ_g, eventually approaching the error of the corrected model. This is because, for higher values of λ_g, the duration of each sleep interval gets shorter, finally approaching zero (Figure 8) and the impact of PSM on the overall power consumption finally becomes negligible.

VI. Related work

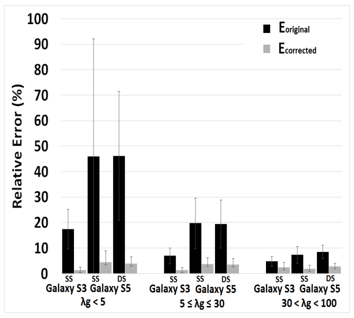
A number of recent works have developed models of the WiFi power/energy consumption in smartphones [1]-[2]. The majority of these works model the NIC power consumption with a finite number of power states (e.g., Tx, Rx, idle, sleep), and add a fixed amount (typically referred to as *base power*) to account for the non-wireless power consumption of the device, ignoring the impact of the packet generation rate. Two notable exceptions (apart from [3]) are PowerTutor [10] and PIMM [2]. PowerTutor models the WiFi active power consumption as a function of both the packet generation rate r_{data} and the PHY bitrate $R_{channel}$: $P = P_{base} + (48 - 0.768 \times R_{channel}) \times r_{data}$. However, the model does not offer any intuition about the cross-factor and does not decouple the WiFi and non-WiFi parts of the power consumption. PIMM, similar to [3], decouples the non-WiFi from the WiFi power consumption and models the former as a function of the packet generation rate.

Additionally, several of these models [11], [12], [13], [14], [15], assume constant power consumption at each power state; this assumption does not hold true for the *active* power state. Shye et al. [1] model the active power consumption as a function of the percentage of the Tx/Rx airtime but ignore other MAC/PHY layer factors such as MCS, number of MIMO streams, channel width, or Tx power. Khan et al. [16] take into account both the airtime and the number of MIMO streams. Other works have considered the impact of signal strength [17], packet size [18], and application layer throughput [19], [20]. In [2], the WiFi active power consumption is modeled as a power function of the total time interval t ($P = a \cdot t^{beta}$), with α, β adjusted for different packet sizes and received signal strength (RSS) levels. Since RSS indirectly accounts for the MCS, one could replace P_{tx} in equation (1) with the power function from [2]. Nonetheless, in this paper, we preferred to work with the original model from [3], as one of our goals was to understand the impact of different WiFi transmission parameters (MCS, NSS, channel width) to the power consumption.

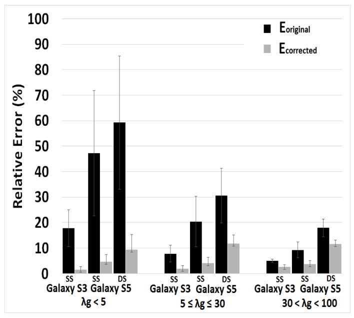
We also note that most of these works focus on the active (plus idle) power consumption and ignore PSM. The authors in [10] observed experimentally that in HTC Dream and HTC Magic phones, WiFi transitions from a high to a low power state when the packet generation rate drops below 8 packets/sec and from the low to the high power state when the packet generation rate is higher than 15 packet/sec but provided no explanation for this behavior. The only works that model the total power consumption as a function of the active, idle, and sleep power, based on the relationship between the data rate and the PSM timeout, similar to our approach, are [15], [18], [2]. [15] assumes a constant active power consumption and [18] models the active power consumption as a function of the packet size only; further, both these works ignore the cross-factor. On the other hand, [2] assumes that the NIC never goes to sleep mode when the packet inter-arrival time is shorter than the PSM timeout, which is not true in modern 802.11n smartphones, as we showed in Section V.

VII. Conclusion

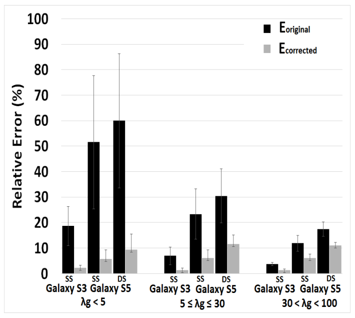
In this paper, we revisited one of most comprehensive WiFi power models [3], initially proposed for wireless APs/routers. We re-evaluated the model for the current generation of smartphones equipped with both 802.11g and 802.11n NICs and found that it remains valid over a range of devices and network types, although the model parameters exhibit different trends compared to the ones reported in the original paper. Further, we looked at how the model parameters are affected by 802.11n's new MAC features. Interestingly, we found that the idle/base power is often the primary contributor to the total power consumption except for very high frame generation rates. Finally, we studied the impact of PSM which was



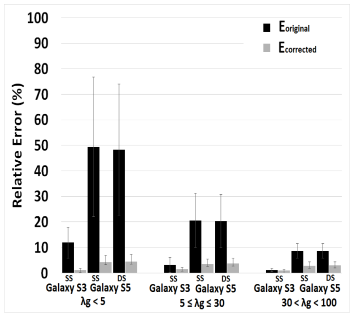
(a) 20 MHz, MCS 1/9.



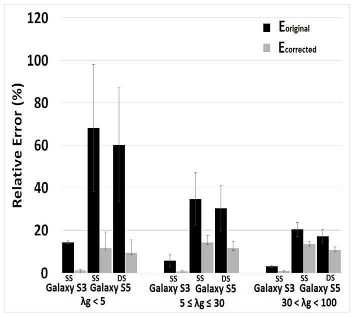
(b) 20 MHz, MCS 4/12.



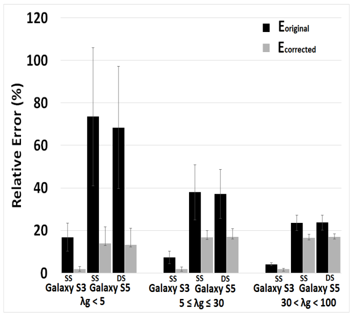
(c) 20 MHz, MCS 6/14.



(d) 40 MHz, MCS 1/9.



(e) 40 MHz, MCS 4/12.



(f) 40 MHz, MCS 6/14.

Fig. 9. Average relative error of the original and the corrected model for Galaxy S3 and Galaxy S5 for three different MCS. The error bars represent the standard deviations.

ignored in the original model and showed that the model fails to accurately estimate the power consumption for low frame generation rates. Based on our observations, we developed a new model that accurately models the power consumption across the full range of frame generation rates and verified its accuracy for a wide range of parameters and devices.

VIII. Acknowledgment

This work was supported in part by NSF grant CNS-1422304.

References

[1] A. Shye, B. Scholbrock, and G. Memik, "Into the wild: Studying real user activity patterns to guide power optimizations for mobile architectures," in *Proc. of ACM Micro*, 2009.

[2] J. Koo, W. Lee, S. Choi, and Y. Park, "PIMM: Packet Interval-Based Power Modeling of Multiple Network Interface-Activated Smartphones," in *Proc. of the ACM 6th International Conference on Future Energy Systems (e-Energy)*, 2015.

[3] P. Serrano, A. Garcia-Saavedra, A. Banchs, G. Bianchi, and A. Azcorra, "Per-frame energy consumption anatomy of 802.11 devices and its implication on modeling and design," *IEEE/ACM Transactions on Networking (ToN)*, vol. 23, no. 4, pp. 1243–1256, 2014.

[4] D. Halperin, B. Greenstein, A. Sheth, and D. Wetherall, "Demystifying 802.11n power consumption," in *Proc. of USENIX Workshop on Power Aware Computing and Systems*, October 2010.

[5] P. M. Yunze Zeng, Parth H. Pathak, "A First Look at 802.11ac in Action: Energy Efficiency and Interference Characterization," in *Proc. of IFIP Networking*, 2014.

[6] S. K. Saha, P. Deshpande, P. P. Inamdar, R. K. Sheshadri, and D. Koutsonikolas, "Power-throughput tradeoffs of 802.11n/ac in smartphones," in *Proc. of IEEE INFOCOM*, 2015.

[7] "ath9k foss wireless driver for atheros ieee 802.11n pci/pci-express based chipsets," http://wireless.kernel.org/en/users/Drivers/ath9k.

[8] N. Warty, R. K. Sheshadri, W. Zheng, and D. Koutsonikolas, "A first look at 802.11n power consumption in smartphones," in *Proceedings of the ACM Mobicom International Workshop on Practical Issues and Applications in Next Generation Wireless Networks (PINGEN)*, 2012.

[9] "Monsoon power monitor." http://www.msoon.com/LabEquipment/PowerMonitor/.

[10] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proc. of the International Conference on Hardware-Software Codesign and System Synthesis (CODES+ISSS)*, October 2010.

[11] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network applications," in *Proc. of ACM/USENIX Internet Measurement Conference (IMC)*, November 2009.

[12] A. Pathak, Y. C. Hu, M. Zhang, V. Bahl, and Y.-M. Wang, "Fine grained power modeling for smartphones using system call tracing," in *Proc. of ACM/USENIX EuroSyS Conference*, April 2011.

[13] J. Manweiler and R. R. Choudhury, "Avoiding the rush hours: WiFi energy management via traffic isolation," in *Proc. of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, June 2011.

[14] R. Mittal, A. Kansal, and R. Chandra, "Empowering developers to estimate app energy consumption," in *Proc. of the 18th Annual International Conference on Mobile Computing and Networking (MobiCom)*, August 2012.

[15] Y. Xiao, Y. Cui, P. Savolainen, M. Siekkinen, A. Wang, L. Yang, A. Yla-Jaaski, and S. Tarkoma, "Modeling Energy Consumption of Data Transmission over Wi-Fi," *IEEE Transactions on Mobile Computing*, vol. 13, no. 8, pp. 1760–1773, 2014.

[16] M. O. Khan, V. Dave, Y.-C. Chen, O. Jensen, L. Qiu, A. Bhartia, and S. Rallapalli, "Model-Driven Energy-Aware Rate Adaptation," in *Proc. of ACM Mobihoc*, 2013.

[17] N. Ding, D. Wagner, X. Chen, Y. C. Hu, and A. Rice, "Characterizing and modeling the impact of wireless signal strength on smartphone battery drain," in *Proc. of ACM SIGMETRICS*, 2013.

[18] J. Li, J. Xiao, J. W.-K. Hong, and R. Boutaba, "FSM-based Wi-Fi Power Estimation Method for Smart Devices," in *Proc. of the IFIP/IEEE International Symposium on Integrated Network Management, (IM)*, 2015.

[19] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A Close Examination of Performance and Power Characteristics of 4G LTE Networks," in *Proc. of ACM Mobisys*, 2012.

[20] F. Xu, Y. Liu, Q. Li, and Y. Zhang, "V-edge: Fast self-constructive power modeling of smartphones based on battery voltage dynamics," in *Proc. of USENIX NSDI*, 2013.