Modeling Energy Consumption of Data Transmission over Wi-Fi

Yu Xiao, Yong Cui, Petri Savolainen, Matti Siekkinen, An Wang, Liu Yang, Antti Ylä-Jääski and Sasu Tarkoma

Abstract—Wireless data transmission consumes a significant part of the overall energy consumption of smartphones, due to the popularity of Internet applications. In this paper we investigate the energy consumption characteristics of data transmission over Wi-Fi, focusing on the effect of Internet flow characteristics and network environment. We present deterministic models that describe the energy consumption of Wi-Fi data transmission with traffic burstiness, network performance metrics like throughput and retransmission rate, and parameters of the power saving mechanisms in use. Our models are practical because their inputs are easily available on mobile platforms without modifying low-level software or hardware components. We demonstrate the practice of model-based energy profiling on Maemo, Symbian and Android phones, and evaluate the accuracy with physical power measurement of applications including file transfer, web browsing, video streaming and instant messaging. Our experimental results show that our models are of adequate accuracy for energy profiling and are easy to apply.

Index Terms—Power modeling, Wi-Fi, smartphone

1 Introduction

Energy consumption caused by wireless data transmission on smartphones is increasing rapidly with the growing popularity of applications that require network connectivity. This results in shrinking battery life, as the development of battery technology is unable to keep up with the energy demand of applications. While waiting for breakthroughs in battery technology, we can try and make the networked applications more energy-efficient.

In order to develop energy-efficient networked applications on smartphones, the developers need to know the factors that affect the energy-efficiency in wireless data transmission and to be able to evaluate the joint effects of these factors on battery life. Although many of these factors, such as the inactivity timers in 3G networks [1], the network throughput [2], and the traffic patterns [3], have been identified through measurement studies, the joint impact of these factors has not been thoroughly quantified. We still lack models that can accurately estimate the data-transmission-related energy consumption of wireless applications in varying network environments.

To remedy the situation, we have built practical power models that utilize traffic characteristics to estimate the energy consumption of Wi-Fi data transmission. Our models can be used for power analysis of network applications, as well as for runtime power estimation in energy-aware applications that utilize technologies such as computation offloading [4], [5] or traffic shaping [6], [7].

We base our models on deterministic power modeling where the basic idea is to estimate the energy consumption of hardware components with the help of predefined state machines. In our case, we have build a state machine that models the standard behavior of an 802.11 WNI. Since the operating systems on most commercial devices do not expose the durations the WNI spends in each power state, we propose using traffic traces to estimate these durations.

The inputs of our models, mainly the traffic statistics such as the burst durations and sizes, are accessible without modifying low-level hardware or software components. While exploring the trade-off between the model accuracy and the granularity of the inputs we find that the burst-level traffic information is enough for power modeling purposes. When high-sampling-frequency power meters are not available, burst-level analysis becomes especially interesting as means of reducing the negative impact of the the low sampling frequency on the accuracy of model-based energy profiling.

Our models are applicable to both TCP and UDP transmission. Due to limited space, we use the more complex TCP transmission in model evaluation. This choice can also be justified by the fact that more than 70% of all IP traffic has been measured to be TCP-based [8]. We evaluate our models with TCP download/upload at different data rates in both fairly and heavily congested networks. Our test cases cover the scenarios where the data is delivered in regularly repeated bursts as often

- *Y. Xiao and M.Siekkinen are with Aalto University, Finland. E-mail: firstname.lastname@aalto.fi*
- *Y. Cui is with Tsinghua University, China.*
- *P. Savolainen and S. Tarkoma are with Helsinki Institute for Information Technology (HIIT) / University of Helsinki and Aalto University, Finland.*
- *A. Wang is with Tsinghua University and Beijing University of Posts and Telecommunications.*
- *L. Yang is with Tsinghua University and Beijing JiaoTong University.*
- *A. Ylä-Jääski is with Helsinki Institute for Information Technology(HIIT) / Aalto University.*

Digital Object Indentifier 10.1109/TMC.2013.51

seen in streaming applications, as well as the scenarios where the data is delivered in bursts with random sizes and intervals as seen in web browsing and instant messaging. We compare the estimation of our models against physical power measurement on Nokia N810, Nokia N95, HTC G1, and Samsung Nexus S. The experimental results show that the Mean Absolute Percentage Error (MAPE) of our estimation varies between 1.1% and 9.1%.

Our contributions presented in this paper include:

- Presenting simple and practical power models of Wi-Fi data transmission based on Internet flow characteristics and network environment context.
- Evaluating the proposed power models through thorough empirical experiments on three mobile platforms and in different network environments.

Compared to the preliminary version of this paper [9], this extended version includes significant new results. We have extended our models to profile the transmission energy consumption of traffic that does not show regular patterns, and to estimate the overhead caused by MAC layer retransmissions in congested networks. Moreover, we have analyzed the impact of the choice of the burst detection threshold and of the the general granularity of the inputs on the resulting power estimation accuracy.

The rest of this paper is organized as follows. Section 2 covers the relevant background of power modeling. Section 3 presents our power models. We discuss the practical issues of model-based energy profiling in Section 4 and present the evaluation of our models in Section 5. The applicability of our models to the emerging 802.11 standards is discussed in Section 6 before we conclude our work in Section 7.

2 Background

Power modeling has been widely used for investigating the factors that influence the energy consumption of smartphones. In this section, we first introduce the two main modeling methodologies used in the discipline, and then introduce the power saving mechanisms that are often used in Wi-Fi network interfaces (WNIs).

2.1 Statistical Power Modeling of Smartphones

Statistical power modeling employs statistical methods such as linear regression for estimating the relationship between the power consumption and some measured variables such as transmission rate or processor clock speed. These methods have been applied in analyzing the power consumption of software components as implemented in PowerScope [10], as well as in modeling the system-level power consumption of the smartphone hardware. Examples of the latter include PowerTutor [11], Sesame [12], and the work presented in [13].

In these system-level power models, power consumption of Wi-Fi data transmission was studied as part of the overall power consumption. We find that the variables used in these models for modeling the Wi-Fi data transmission only provide coarse-grained information, such as uplink channel rate and network throughput, which cannot well describe the impact from traffic patterns and the network environment. For example, according to [11], PowerTutor estimates the power consumption of the WNI using the model shown in (1).

$$P_{Wi-Fi} = P_{baseline} + (48 - 0.768 \times R_{channel}) \times r_{data}, \quad (1)$$

where $P_{baseline}$ is the baseline power corresponding to the power state of the WNI, $R_{channel}$ is uplink channel rate ranging from 1Mbps to 54Mbps, and r_{data} is packet rate. The WNI is assumed to switch between two power states according to the packet rate of the Wi-Fi data transmission. Except $P_{baseline}$ which is constant and hardware-specific, $R_{channel}$ and r_{data} are collected from the phones during runtime.

2.2 Deterministic Power Modeling of Smartphones

From software point of view, a hardware component can conduct different operations, each of which corresponds to an operating mode. For example, a WNI has at least three operating modes, corresponding to the operations of sending, receiving and waiting for traffic, respectively. In most cases where one operating mode corresponds to exactly one power state, the power consumption of the hardware component can be derived from the operating mode, and vice versa.

There are also exceptions where the hardware components auto adapt their operation to their current workload, and thus something the software sees as one single operating mode can in fact include several hardware power states. For example, Pathak et al. [14] observed that on HTC Tytn2 running Windows Mobile 6, the WNI can switch to a power state with higher power consumption when the packet rate gets over 50 packets per second. Another example is the 802.11 Power Saving Mode (PSM) [18] which will be described in Section 2.3.

Deterministic power models describe the power consumption behavior of a hardware component with a power state machine. The total energy cost of a hardware component over time is composed of the energy that the component spends in each of its power states and of the energy spent during the transitions between the power states. It can be formally presented as follows.

$$E(t) = \sum_j E_j(t_j) + \sum_j \sum_k E_{j,k} \times C_{j,k}(t), \quad (2)$$

where $E(t)$ is the total energy consumed by the hardware component over the duration t, $t = \sum_j t_j$, t_j is the duration spent in power state j and $E_j(t_j)$ is the energy spent during t_j. Assuming that P_j, the rate of energy consumption in power state j, is constant during t_j, $E_j(t_j)$ can be calculated as the product of t_j and P_j. $E_{j,k}$ is the overhead caused by the transition from power state j to k, while $C_{j,k}(t)$ shows how many times this transition has occurred during t.

Deterministic power modeling has been used for studying energy consumption of Wi-Fi [15], 3G [16],

LTE [17] and Bluetooth [2]. The operating mode of a hardware component can be tracked using three methods. First is to directly read the information about the operating mode from the hardware component via a device driver of the OS. For instance, Quanto [19], a network-wide energy profiler for embedded network devices, adopts this method. However, as standard device drivers do not usually expose the operating mode information, Quanto requires modifications to device drivers.

Second is to estimate the operating mode based on system call traces, as proposed by Pathak et al. [14]. However, the smartphones must be flashed with customized kernel images to enable the system-call tracing.

Third is to derive operating mode from measured workload. For example, the workload of network transmission can be described with *libpcap*[1] packet traces. These traces can tell if the WNI is sending, receiving or waiting for packets. Moreover, they can provide traffic statistics, such as throughput and packet rate, which are useful for detecting workload-driven power state transition. In practice, a power state machine of the WNI can be built by empirically correlating changes in the packet traces to physically measured changes in power levels. With the help of such a state machine, t_j and $C_{j,k}(t)$ could be derived from a *libpcap* packet trace. We adopted the third method in this work.

2.3 Wi-Fi transmission cost

The total energy consumption of Wi-Fi data transmission consists of two parts, the energy consumption of the WNI and that of the CPU and memory during data copying and processing operations [20]. According to our measurement on Samsung Nexus S, data copying operations consumed up to 15% of the overall energy consumption during Wi-Fi data transmission. In this paper we focus on the former part, also called *the transmission cost*.

The power state machine of a hardware component includes the state transitions defined by the power saving mechanisms in use. An 802.11 WNI has three default operating modes, namely, TRANSMIT, RECEIVE and IDLE. The 802.11 PSM [18] introduces another operating mode called SLEEP. When the WNI stays in the SLEEP mode, the WNI only wakes up at a granularity of beacon intervals (e.g. 100ms) to check for incoming traffic. As a result, it costs much less energy than in any other mode. However, it may cause performance degradation, because the traffic that arrives between beacons is either buffered at the access point or simply dropped if the buffer overflows. To solve this issue, an adaptive version of PSM, also known as PSM Adaptive, has been proposed and widely adopted in commercial products. In PSM Adaptive, after receiving or transmitting a packet, the WNI will stay in the IDLE mode for a period of time before going to sleep. We call the length of this period

1. *libpcap* is a portable C/C++ library for network traffic capture. It is available on www.tcpdump.org.

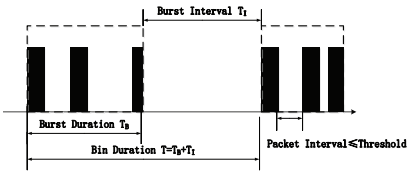


Fig. 1. Burst definition.

the PSM timeout, whose default value varies from device to device. In the remainder of this paper, we will use the abbreviation PSM to refer to PSM Adaptive.

3 Power Modeling

Using the method outlined in Section 2.2, we propose modeling the Wi-Fi transmission cost using easily-accessible traffic information. As presented in [21], Internet traffic is bursty on a small, typically less than 100-1000ms scale. We utilize this burstiness phenomenon in estimating the power state of the WNI, and build power models that define variables with burst size/duration, and data rate. The definition of traffic burst is given in Section 3.1.

We model a complete TCP/UDP session as a combination of upstream and downstream traffic. We first study the power consumption of one way traffic (Section 3.2), and then explain how the combination of the developed models can be used for modeling power consumption of TCP download and upload sessions (Section 3.3). We provide simplified forms of the models in Section 3.4 and discuss the energy consumption overhead caused by MAC layer retransmission in Section 3.5.

Two important metrics used in this paper are *Power* and *Energy Utility*. Power is the energy consumption per unit time expressed in Watts, while Energy Utility is the average throughput per unit energy [22]. The notation used in this section is summarized in Table 1.

3.1 Burst Definition

An Internet flow can be considered as a train of packets. According to the definition of "train burstiness" in [23], "a burst can be defined as a train of packets with a packet interval less than a threshold θ". An Internet flow can then be divided into bins with one burst in each bin. One burst includes one or more packets, depending on the distribution of packet intervals and the value of θ.

Due to the difference in Power between the TRANSMIT and the RECEIVE modes, we add one constraint to the definition of "train burstiness" [23]. We define a burst as a train of packets with the same transmission direction and with each packet interval smaller than the threshold θ. As shown in Fig. 1, burst duration T_B is "the time elapsed between the first and the last packets of a burst" [23], while burst size S_B is the amount of the data sent or received during T_B. Burst interval T_I is the time elapsed between the last packet of a burst and

TABLE 1
Summary of notation

S_B	burst size	T_B	burst duration	T_I	burst interval
E_B	energy cost of a bin	$E_0(r)$	energy utility at data rate r	T	bin duration
r	bin data rate	P_{active}	average power in active mode	P_S	power in the SLEEP mode
P_R	power in RECEIVE mode	P_T	power in TRANSMIT mode	P_I	power in the IDLE mode
$P_d(r)$	power of downlink at data rate r	$P_u(r)$	power of uplink at data rate r	r_c	threshold of data rate
E	energy cost of an Internet flow	T_{sleep}	time spent in SLEEP mode within a bin	T_d	total duration of downlink bursts
T_u	total duration of uplink bursts	S_{db}	downlink burst size	S_{ack}	size of ACK
T_s	total duration in SLEEP mode	$T_{timeout}$	the value of the PSM timeout	$E_{retransmit}$	cost of retransmitting packets
R_r	retransmission ratio	T_{ir}	retransmitted packet interval	$\overline{r}$	throughput over a flow
r_{max}	maximum processing capacity of downstream traffic of WNI	θ	maximum packet interval within a burst		

the first packet of the following burst. Bin duration T includes the burst duration and the burst interval.

Given an Internet flow, we can detect all the bursts and then use the burst information to calculate the average network throughput $\overline{r}$ over the Internet flow following

$$\overline{r} = \frac{\sum S_B}{\sum T} = \frac{\sum S_B}{\sum T_B + \sum T_I}. \quad (3)$$

From (3) we can see that given a fixed amount of data and a fixed data rate limit, the data can be delivered in different traffic patterns in terms of distributions of burst size and interval. We use the standard deviation of burst interval and that of burst size to describe the regularity of the bursts. If Internet flows consist of bursts with small standard deviations, such as those caused by audio streaming, we consider these flows to be regularly bursty traffic and to be randomly bursty traffic otherwise. We will describe the power models that fit these two kinds of traffic in Section 3.2.2.

3.2 Downlink/Uplink Power Consumption

According to our definition of train burstiness, a downlink or an uplink flow can be divided into bins. We propose aggregating the energy spent in each bin (Section 3.2.1) into the transmission cost of a flow (Section 3.2.2).

We assume that the threshold value θ is always smaller than the PSM timeout $T_{timeout}$ when the PSM is enabled. This means that the transition from the IDLE to the SLEEP mode may only happen during burst intervals. Let T_{sleep} be the duration spent in the SLEEP mode during a burst interval. As described in (4), only when the value of T_I is greater than that of $T_{timeout}$ can the WNI switch to the SLEEP mode. Let r be bin data rate. In (5) we define a threshold r_c as the bin data rate when T_I is equal to $T_{timeout}$.

$$T_{sleep} = T_I - T_{timeout}, when\ T_I > T_{timeout}. \quad (4)$$

$$r_c = \frac{S_B}{T_B + T_{timeout}}. \quad (5)$$

To evaluate the effect of the PSM, we define the following two scenarios. The WNI is expected to always stay in the IDLE mode during burst intervals in Scenario 1. Thus only in Scenario 2 can the PSM save energy.

- *Scenario 1*: PSM is disabled, or r is not smaller than r_c with PSM enabled.
- *Scenario 2*: r is smaller than r_c with PSM enabled.

3.2.1 Energy per bin

We denote by P_T, P_R, P_I and P_S the Power when the WNI stays in the TRANSMIT, RECEIVE, IDLE and SLEEP mode, respectively. As some modern smartphones may support transmit power control, we make the simplifying assumption that the transmit power stays the same within one burst and only can change between the bursts. We estimate the Power within one burst to be fixed to either P_T or P_R, depending on the transmission direction. Our estimation ignores the transition into the IDLE mode during the packet intervals smaller than the threshold value θ. The potential error caused by it will be analyzed in Section 4.

Downlink power consumption is the power consumed when receiving data. Let E_B denote the transmission cost of a bin in Joule, and $P_d(r)$ denote the average downlink Power in Watts. In Scenario 1, the WNI operates in the RECEIVE mode when receiving data and in the IDLE mode otherwise. Thus E_B includes the energy spent in the RECEIVE and the IDLE modes. In Scenario 1, the value of $P_d(r)$ can increase linearly with the bin data rate r, as shown in (6).

$$P_d(r) = \frac{E_B}{T} = \frac{P_R T_B + P_I T_I}{\frac{S_B}{r}} = P_I + r\frac{T_B}{S_B}(P_R - P_I). \quad (6)$$

In Scenario 2, T_I is divided into two parts, $T_{timeout}$ and T_{sleep}. The WNI can be in the IDLE mode for a duration of $T_{timeout}$ after receiving the last packet of data, and in the SLEEP mode after this until the end of the bin. E_B can then be divided into 3 parts as shown in (7). Accordingly, the definition of $P_d(r)$ is refined into (8).

$$E_B = P_R T_B + P_I T_{timeout} + P_S T_{sleep}. \quad (7)$$

$$P_d(r) = P_s + r[\frac{T_B}{S_B}(P_R - P_S) + \frac{T_{timeout}}{S_B}(P_I - P_S)]. \quad (8)$$

3.2.2 Power over an Internet flow

If the bursts included in an Internet flow are regularly repeated, S_B and T_B can be considered to be fixed, while

the length of the burst interval T_I varies with the bin rate r, for example, T_I increases when r decreases[2]. In that case, the Internet flow can be compared to one single bin that repeats itself over and over again for the whole duration of the flow. Thus (6) and (8) can be used for estimating the average power over the Internet flow by replacing r with the $\overline{r}$ defined in (3).

According to (6) and (8), Power increases linearly with data rate for regularly bursty traffic. We denote the Energy Utility of the Internet flow by $E_0(\overline{r})$ and define it in (9). Similarly with Power, we can see that $E_0(\overline{r})$ increases with $\overline{r}$, which means it is more energy-efficient to transfer regularly bursty traffic at a higher rate.

$$E_0(\overline{r}) = \frac{\overline{r}}{P_d(\overline{r})}. \tag{9}$$

If the bursts included in an Internet flow are not regularly repeated, which means the burst sizes and intervals vary over time, the total energy consumption E can be aggregated from the energy spent in each bin. When the PSM is disabled, E and $P_d(\overline{r})$ can be calculated following (10) and (11) respectively.

$$E = \sum E_B = \sum T_B P_R + \sum T_I P_I. \tag{10}$$

$$P_d(\overline{r}) = \frac{E}{\sum T} = P_R - \frac{\sum T_I}{\sum T}(P_R - P_I). \tag{11}$$

When the PSM is enabled, let T_S denote the total duration spent in the SLEEP mode, and c be the total number of burst intervals. $P_d(\overline{r})$ in this case can be obtained from (12).

$$P_d(\overline{r}) = P_R - \frac{\sum T_I - T_S}{\sum T}(P_R - P_I) - \frac{T_S}{\sum T}(P_R - P_S), \tag{12}$$

where,

$$T_S = (\sum_{T_I > T_{timeout}} T_I) - cT_{timeout}(1 - \sum_{T_I \leq T_{timeout}} T_I)). \tag{13}$$

The above equations (10) - (12) can be applied for estimating average uplink Power $P_u(\overline{r})$ by replacing $P_d(\overline{r})$ with $P_u(\overline{r})$, and P_R with P_T.

3.3 TCP Download/Upload Power Consumption

We model TCP transmission as a combination of separate downlink and uplink transmissions. Let r_d be the downlink data rate, and r_u be the uplink data rate. Take TCP download as example, r_d is the data rate of downloading the files, while r_u is the data rate of sending ACKs.

We first discuss the power consumption of TCP download. We assume that a downlink burst includes n packets, and is followed by uplink bursts that consists of in total m ACKs.[3] Let the downlink burst size be S_{db} and the size of one ACK be S_{ack}. The uplink data rate r_u can be obtained from (14).

$$r_u = \frac{mS_{ack}r_d}{S_{db}}. \tag{14}$$

We extend the definition of a bin here to have a bin including one downlink burst and all the uplink bursts sent before the beginning of the next downlink burst. The bin duration is the duration from the first packet in the downlink burst until the first packet of the next downlink burst. We assume that the downlink and uplink bursts do not overlap.

We denote the downlink burst duration by T_d, and the uplink burst duration by T_u. In the cases of TCP download/upload, we redefine the threshold of network throughput r_c in (15). Having the data rate smaller than r_c is a necessary condition for the WNI to go to sleep during a bin. Whether the WNI will go to sleep and how many times the WNI will switch into the SLEEP mode within a bin depends on each value of the burst intervals within the bin.

$$r_c = \frac{S_{db}}{T_d + T_u + T_{timeout}}. \tag{15}$$

Let the average Power during the TCP download be $P(r_d)$. It consists of both downlink and uplink power. In the Scenario 1 defined in Section 3.2, $P(r_d)$ can be calculated based on (6), as follows.

$$\begin{aligned} P(r_d) &= P_d(r_d) + P_u(r_u) - P_I \\ &= P_I + \frac{r_d T_d}{S_{db}}(P_R - P_I) + \frac{r_u T_u}{mS_{ack}}(P_T - P_I) \\ &= P_I + \frac{r_d}{S_{db}}[T_d(P_R - P_I) + T_u(P_T - P_I)]. \end{aligned} \tag{16}$$

In the Scenario 2 defined in Section 3.2, $P(r_d)$ can be estimated following (17).

$$\begin{aligned} P(r_d) = P_s + \frac{r_d}{S_{db}}[&T_d(P_R - P_S) \\ &+ T_u(P_T - P_S) + \alpha T_{timeout}(P_I - P_S)], \end{aligned} \tag{17}$$

where $\alpha T_{timeout}$ is the total duration the WNI will spend in the IDLE mode during all the burst intervals within a bin. The factor α is calculated as follows: Assume that there are X burst intervals within the bin, out of which Y intervals are longer than $T_{timeout}$. In the beginning of each of these Y intervals the WNI stays in the IDLE mode for the duration of $T_{timeout}$ before going to sleep. Additionally, the WNI stays in IDLE mode during the complete duration of the $X - Y$ intervals that are shorter than $T_{timeout}$. The factor α is thus:

$$\alpha = Y + \frac{1}{T_{timeout}} \sum_{i=1}^{X-Y} T_i : T_i < T_{timeout}. \tag{18}$$

Similarly with the TCP download, the power consumption of the TCP upload can be calculated as presented in (16) and (17) by replacing r_d with r_u, and S_{db} with the data size of the uplink data burst. When considering the power consumption of multiple TCP

2. Keeping the burst size and burst duration constant and varying the length of the burst interval according to the desired network throughput is a data-rate-limiting mechanism utilized in many traffic shaping utilities such as Trickle [24]

3. Depending on the TCP version, there may be one ACK for each received packet, or one ACK for multiple received packets. Depending on the intervals between ACKs and the threshold value in the burst definition, the ACKs may be divided into more than one uplink burst.

TABLE 2
Different forms of power models

Information required	Parameters	Eq.	Accuracy
packet size, arrival time, transmission direction	burst size, duration, interval, and transmission direction	(16) (17)	high
packet arrival time, transmission direction	burst duration, transmission direction	(20)	high
throughput	throughput	(21)	low

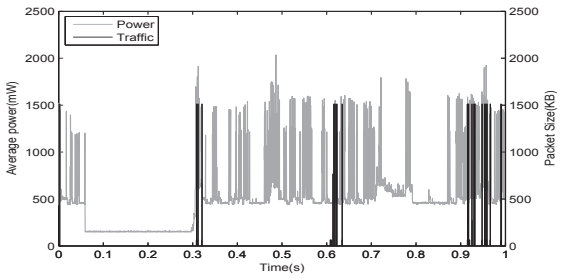


Fig. 2. Synchronized I/O graph and power consumption on Nexus S. The device backlight was turned off.

connections, the aggregate network data rate has to be taken into consideration. In practice, we replace the r_d and r_u in (16) and (17) with the aggregate data rate in each direction. The extra protocol processing cost of multiple TCP connections can be ignored when compared to the uplink and downlink transmission cost.

3.4 Simplified Power Models

As listed in Table 2, the models presented in Section 3.3 require information including packet size, arrival time and transmission direction. In this section we provide two simplified power models that require less information for the Scenario 1 defined in Section 3.2.

The first one is to estimate the average Power over the Internet flow based on the average Power in active mode. Here we define *active mode* as the operating mode of the WNI when it stays in either the TRANSMIT or the RECEIVE mode. We denote the average Power in active mode by P_{active}. It can be calculated based on the durations of uplink and downlink bursts as shown in (19). The average Power over the Internet flow can then be transformed from (11) into (20) by replacing P_R with P_{active}. (20) can be applied to any traffic pattern and can be applied for both TCP/UDP download and upload.

$$P_{active} = \frac{\sum T_u \times P_T + \sum T_d \times P_R}{\sum T_u + \sum T_d}. \quad (19)$$

$$P = P_{active} - \frac{\sum T_I}{\sum T} \times (P_{active} - P_I). \quad (20)$$

The second one is to simplify the power models by ignoring ACKs. Due to the small sizes of ACKs, receiving/sending an ACK in a modern smartphone usually costs less than 1ms. The energy cost of sending ACKs is so small compared to the cost of transmismitting data packets. Thus the energy cost of ACKs can be dropped from (16) and (17) for practical usage if a higher error rate is acceptable. In addition, the packet intervals in each burst are limited by the threshold θ. If we assume that the packet intervals can be ignored, the data rate of a downlink burst can be considered to be equal to the maximum processing capacity of downlink traffic of the WNI. We denote it by r_{max}. When the PSM is disabled, (16) can be simplified into (21).

$$P(r_d) = P_I + \frac{r_d}{r_{max}}(P_R - P_I). \quad (21)$$

To calculate the energy consumption of TCP upload, replace P_R with P_T, r_d with r_u, and r_{max} with the maximum processing capacity of uplink traffic in (21).

3.5 MAC layer retransmission

From energy viewpoint, retransmitting a packet is not different from transmitting a "fresh" packet. We ran Wireshark[4] on Samsung Nexus S while sending packets in a congested network, and then synchronized the traffic trace with the power measurement trace. As shown in Fig. 2, the black line represents the I/O graph with each spike corresponding to one packet captured by the Wireshark running on the phone[5]. The gray line shows the power consumption of the phone during data transmission. We can find a lot more spikes in gray color each of which corresponds to a retransmitted packet.

The overhead caused by MAC layer retransmission includes two parts. One is the energy spent in retransmitting packets on the sender side. According to the retransmission mechanisms used in 802.11 [18], the sender may retransmit a packet several times until the transmission succeeds or until the retransmission limit is reached. Let T_{ir} be the interval between a retransmitted packet and its previous packet. If the value of T_{ir} is not greater than the threshold value θ and the previous packet is an uplink packet, the retransmitted packet can be considered to be part of the uplink burst the previous packet belongs to. In other words, the uplink burst duration is increased by T_{ir}, due to retransmission. We define $E(T_{ir})$ as the expected value of T_{ir}, and $\#(retransmit)$ as the total number of retransmitted packets. The cost of retransmitting packets $E_{retransmit}$ can be calculated as below.

$$E_{retransmit} = \#(retransmit) \times E(T_{ir}) \times (P_T - P_I). \quad (22)$$

Given a packet trace captured on network layer, we denote its packet count by $\#(packet)$. The value of $\#(retransmit)$ can be calculated as below.

$$\#(retransmit) = \#(packet) \times \frac{R_r}{1 - R_r}, \quad (23)$$

4. www.wireshark.org

5. The black line does not include MAC layer retransmission, since monitoring of retransmission at MAC layer requires the WNI to run in monitor mode, whereas the WNI cannot be used for transmitting or receiving data while operating in monitor mode

where R_r is the retransmission ratio calculated from MAC layer traffic information. For example, if we capture N packets on MAC layer including M retransmission attempts, the value of R_r is equal to $\frac{M}{N}$.

The other part of the retransmission overhead is caused by the increase in the baseline cost, due to the Dynamic Voltage Frequency Scaling (DVFS) mechanism of CPUs. The basic idea of DVFS is to adapt the CPU frequency to the processing workload. The extra workload caused by retransmission may lead to an increase in the CPU frequency, with the result that the baseline cost represented by the values P_T, P_R, P_I and P_S increases accordingly. For example, in Fig. 2, the Power in the IDLE mode during 0.1s and 0.3s is only 0.177W, whereas it gets close to 0.5W during the interval between 0.8s and 0.85s. Meanwhile, the Power while sending packets increases by around 0.3W when the retransmission starts. This change in Power is consistent with the change in the CPU frequency from 100MHz to 200MHz. Hence, to estimate the transmission cost in congested networks, fine-grained CPU frequency measurement is necessary for providing the right inputs for the power models.

4 Practical Issues

The power models presented in Section 3 require two different kinds of inputs. The first type are the hardware parameters such as P_T and P_R, while the other type represent the burst information derived from traffic traces. In this section we describe the methods of obtaining the values of these inputs, and analzye the impact of these values on the resulting estimation accuracy of our power models. We will leave the errors of power meter calibration out of the scope of this work, and will focus on the errors that come from the processing of the readings obtained from power meters.

4.1 Measurement of Hardware Parameters

We measured the values of P_T and P_R and the maximum throughput of the WNI while sending/receiving back-to-back packets. First, we sent fixed-size packets as fast as possible from the phone at a given CPU frequency, and used the average power over a period (e.g. 60 seconds) as the corresponding value of P_T. Second, we calculated the maximum uplink throughput as the packet size divided by the mean of the measured packet intervals. Similarly, we measured P_R and the maximum downlink throughput while receiving packets to the phone.

We measured the power in the IDLE mode of the WNI, P_I, by first transmitting packets from the device, and then measuring the power after stopping the transmission. On devices with the 802.11 PSM feature, the WNI went to the SLEEP mode after the PSM timeout had passed from the stopping of the transmission, and we could then measure the value of P_S. On the devices with the DVFS feature, such as Samsung Nexus S, the hardware parameter values needed to be measured for all the possible CPU frequencies.

The way we measure P_T and P_R is based on the assumption that the WNI always stays in either the TRANSMIT or the RECEIVE mode during back-to-back packet delivery. However, due to the delays caused by the data transmission over the air and the packet processing, true back-to-back packet delivery of packets is not possible. This means that there always will be small intervals between the packets, and it is possible that the WNI may switch into the IDLE mode during these small intervals. Because the power consumption in the IDLE mode is lower than that in the TRANSMIT or the RECEIVE modes, the measured power may be lower than the correct value would be.

In order to analyze this measurement error in the TRANSMIT case, we denote the measured value of P_T with $P_T^{'}$, the duration of the measurement with T_m, and the actual duration spent in the IDLE mode during T_m with T_e. Now, the proportion $d = \frac{T_e}{T_m}$ gives an estimate of the measurement error. Further, the relationship between the correct P_T and the measured $P_T^{'}$ can be expressed as

$$P_T^{'} = \frac{P_T(1-d)T_m + P_I d T_m}{T_m} = P_T - d(P_T - P_I). \quad (24)$$

Similiar analysis can also be applied in the RECEIVE case, but the equations are omitted due to space restrictions.

4.2 Error in Power Estimation of Bursts

When using our power models for runtime power estimation, we make the assumption that the WNI always stays in the TRANSMIT mode during uplink bursts, and in the RECEIVE mode during downlink bursts. This assumption is similar to the assumption that the WNI always stays in either the TRANSMIT or the RECEIVE mode during back-to-back packet delivery, which we made when measuring the hardware parameters. The similarity of the assumptions implies that they might also carry a similar error source, and this is indeed the case. Even within a burst, there are intervals between the packets, and during these intervals the WNI may enter the IDLE mode.

We now analyse this error in the case of uplink bursts, but the analysis can be trivially applied to downlink bursts as well. The actual energy consumption of WNI during the total duration of uplink bursts T_u consists of two parts, the energy spent in the TRANSMIT mode and the energy spent in the IDLE mode during the packet intervals. We denote the proportion of the time spent in the IDLE mode to T_u with $d^{'}$. The actual Power during uplink bursts can be calculated following (25).

$$P = \frac{P_T(T_u - d^{'}T_u) + d^{'}T_u P_I}{T_u} = P_T - d^{'}(P_T - P_I). \quad (25)$$

Because we use $P_T^{'}$ as the value of P_T in our model-based power estimation, by comparing the values of $P_T^{'}$ and P we can get an estimate of the error in power

estimation of uplink bursts. We denote this error with e_u, and it can be calculated following (26).

$$e_u = P - P_T^{'} = (d - d^{'})(P_T - P_I). \quad (26)$$

Similar analysis can also be applied in analyzing e_d, the error in power estimation of downlink bursts, by replacing P_T with P_R and $d^{'}$ with the proportion of the time spent in the IDLE mode to T_d in (26).

4.3 Impact of the Threshold Value

A packet trace can be divided into different number of bursts with different packets included in each burst, depending on the value of the threshold θ that is used for burst detection. The distribution of packet intervals within bursts and further the value of $d^{'}$ vary with the value of θ. In an extreme case where the value of θ is 0, each burst only includes one packet and the value of $d^{'}$ is 0. The probability of getting a higher value of $d^{'}$ increases with the value of θ. According to (26), if we increase the value of θ starting from 0, the values of e_u and e_d will first decrease until the value of $d^{'}$ becomes equal to that of d. After that, the values increase with θ.

We define the MAPE of the power estimation of a flow as below.

$$MAPE = \frac{|P_m - P_e|}{P_m} = \frac{|e_u T_u + e_d T_d|}{P_m T_{flow}} \quad (27)$$

where P_m is the measured power, P_e is the estimated power, and T_{flow} is the flow duration. The burst durations, T_u and T_d, do not decrease with the value of θ. However, the values of e_u and e_d may decrease with θ, if θ is small enough. In that case, the value of MAPE would increase or not, depending on whether the increase in T_u and T_d can overtake the decrease in e_u and e_d. When θ is big enough, the value of MAPE will always increase with θ.

To evaluate these features, we take TCP download/upload on Samsung Nexus S as an example. Given the same set of packet traces and power measurement results, we tune the value of θ and compare the resulting estimation accuracy. As shown in Fig. 3, the value of MAPE first decreases together with the value of θ, and then increases. The threshold value of 5ms provides generally the best accuracy for all the three cases. According to our experience[6], given the same values of the hardware parameters, a threshold value chosen to be optimal for one of the test cases (e.g. concurrent TCP flows) usually also provides reasonable good accuracy in the other test cases as well.

5 Evaluation

We chose TCP transmission as an example and evaluated the power models presented in Section 3 with physical power measurement on four smartphones, Nokia N810, Nokia N95, HTC G1 and Samsung Nexus S. As listed in Table 3, we evaluated our models in TCP transmission scenarios at various data rates with various traffic patterns and in different network environments. More specifically, we first conducted the experiments in an ideal network environment where the processing latency and packet loss can be ignored. We measured the power consumption of TCP download/upload with the data rate ranging from 16KBps to over 2MBps, and repeated the measurement with the data transmission using a number of separate TCP flows. After that, we conducted TCP download/upload experiments in a congested network environment, with the retransmission ratio varying between 10% and 30%. We also compared our estimation with the physical measurement of four real Android applications. The traffic generated by these Android applications had different characteristics, in terms of both traffic size and pattern. The experimental setup and the results of each test case are explained in this section.

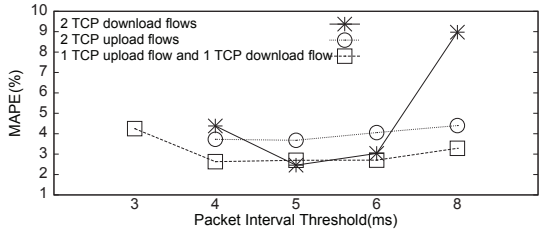


Fig. 3. Comparison of MAPE with different threshold values in the test case 4) defined in Table 3.

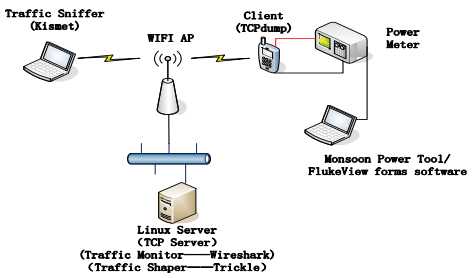


Fig. 4. Experimental setup.

5.1 Experimental Setup

Our experimental setup is illustrated in Fig. 4, including TCP download/upload setup, and power measurement and traffic capturing tools.

5.1.1 TCP download/upload setup

We used an open source TCP utility, netcat[7], as the TCP server running on a Linux server, and our own custom netcat-based mobile applications as the TCP clients. To avoid the energy cost of data copy operations, the downloaded files were written to $/dev/null$ instead of phone memories, and to the same end, the TCP clients

6. When evaluating the power models for Samsung Nexus S, we set the value of θ to be 5ms for all the test cases.

7. http://netcat.sourceforge.net/

TABLE 3
List of test cases

No.	Description	No.flows	Data rate limit	TCP window scaling	Phone models	Equations
1)	TCP download/upload with PSM enabled	1	Enabled	Disabled	N810, N95	(17)
2)	TCP download/upload	1	Enabled	Disabled	N810,N95 HTC G1	(16)
3)	TCP download/upload	1	Enabled/Disabled	Enabled	Nexus S	(20)
4)	Concurrent TCP download/upload	2/4/8	Enabled/Disabled	Enabled	Nexus S	(20)
5)	TCP download/upload in congested network	1	Enabled/Disabled	Enabled	Nexus S	(22)
6)	Web browser, Fengxing(video streaming), Dropbox(file upload), QQ(instant messenger)	1	Disabled	Enabled	Nexus S	(16),(17), (21)

TABLE 4
Experimental setup

Phone model	OS	Power meter	Sampling frequency
Nokia N810	Maemo 4.1	Fluke 189 logging multimeter	20Hz⋆
Nokia N95	Symbian S60	Nokia Energy Profiler	4Hz
HTC G1	Android 1.6	Fluke 189 logging multimeter	20Hz⋆
Samsung Nexus S	Android 2.3	Monsoon Power Monitor[9]	5000Hz

⋆ The logging interval in FlukeView is 1 second. The system logs the average of the 20 samples taken during each second.

TABLE 5
Parameter values for N810, N95 and G1

Phone model	Display	P_T(W)	P_R(W)	P_I(W)	P_S(W)
Nokia N810	off	1.258	1.181	0.884	0.042
Nokia N95	off	1.687	1.585	1.038	0.088
HTC G1	off	1.097	0.900	0.650	-

TABLE 6
Parameter values for Nexus S

CPU frequency	Display	P_T(W)	P_R(W)	P_I(W)	Data rate (KBps)
100MHz	off	1.094	0.867	0.177	$\leq$ 128
100MHz	off	1.130	0.903	0.213	160 - 256
200MHz	off	1.245	1.021	0.435	$\geq$ 512
100MHz	on	1.217	0.887	0.742	512
200MHz	on	1.376	1.050	0.800	1024
400MHz	on	1.549	1.208	0.890	$\geq$ 1536

on the phones generated all the data they uploaded on-the-fly instead of reading it from mass memory.

The TCP clients always try to send/receive data as fast as possible if no data rate limit is applied. In that case, the actual throughput is limited by the available network bandwidth and the processing capacity of the smartphones. To conduct data transmission at a fixed rate, we ran Trickle [24], an utility for bandwidth throttling, on the same Linux server with the TCP server. In addition, apart from Nexus S, the TCP window scaling option was disabled and the MTU was set to 1500 Bytes, so that the protocol processing cost for each packet could be considered to be fixed [25]. Moreover, disabling the TCP window scaling can increase the probability that the transmission rate would be limited by the receiver window, and hence, maximum window size would be used and that would generate bursts of equivalent size.

5.1.2 Power measurement

As described in Table 4, apart from Nokia N95, for which we used the power measurement software, Nokia Energy Profiler[8], we connected all the other tested phones to an external DC power supply and used physical power meters to measure the instant power consumption. The readings of the physical meters were logged from the PC software provided by the manufacturers.

We measured the power consumption with the WNI operating in different states, as described in Section 4.1. The results are listed in Tables 5 and 6. During the measurements, only the basic components of the devices were in use. We also note that the measured values of P_T and P_R include the cost of network protocol processing.

Because the Android phones we used did not provide any interface for adjusting PSM parameters, we measured them using the default settings. The measurement results seem to fit the "PSM disabled" version of our power models. Therefore, we did not provide the value of P_S for these Android phones. In addition, we observed from Nexus S that its CPU frequency varied with the transmission rate. For instance, when the display was turned off, the CPU frequency increased from 100MHz to 200MHz whenever the data sending rate of the phone increased from 256KBps to 512KBps[10]. Therefore, we list in Table 6 the parameter values separately for each CPU frequency. The listed data rate information is to show which values were used in our calculations. They are not necessarily the exact thresholds used in DVFS.

5.1.3 Traffic capturing

The smartphones were connected to 802.11g access points (APs) with beacon intervals set to 100ms. On the phones that support fine-tuning the PSM settings, Nokia N810 and N95, the PSM timeout was set to be 100ms.

During the experiments of TCP download/upload, the network traffic was monitored by running Wireshark on the Linux server that the smartphones connected to.

8. http://www.developer.nokia.com/Resources/Tools_and_downloads/Other/Nokia_Energy_Profiler/

10. Due to the partial wake up mechanism in Android, the CPU worked at a reduced frequency when the display was turned off.

Except in the test case 5), the cross traffic in our APs was assumed to be insignificant. In the test case 5), we ran Kismet[11] on a separate laptop to capture MAC layer traffic information from the WNI operating in the monitor mode. The captured information was later used for calculating MAC layer retransmission rate and the expected value of the packet retransmission interval.

5.2 TCP download/upload with a Single Flow

We first evaluated our models with regularly repeated bursty traffic. On N810, N95 and G1, we repeated TCP download/upload with data rate ranging from 16KBps to 256KBps. The traffic was shaped by Trickle into regular bursts with size of 4KB. As N810 and N95 supports PSM, we measured their power consumption with PSM enabled and disabled, respectively. The measured values were compared with the model-based estimation from the traffic captured from the Linux server.

We empirically set the threshold value for burst detection in our models to be between 6 and 10ms, in order to minimize the standard deviation of the detected burst sizes. This allowed us to use 4KB as the data burst size estimate for both TCP download and TCP upload. We calculated the average downlink and uplink burst durations over the detected bursts. The burst durations are listed in Table 7. For N810 and N95 that support PSM, the burst size and duration information was used for calculating the threshold value of the network data rate r_c, as described in (15). The value of r_c was 37KBps for N810 and 36KBps for N95 in both TCP download and upload cases.

We estimated the transmission cost from the burst information and the parameter values listed in Table 5. The estimated values were calculated following (16) or (17), depending on which conditions of the two scenarios defined in Section 3.2 were satisfied. The value of α in (17) was set to 2 when estimating the power consumption of N95 at the data rate of 16KBps[12]. In all other cases, it was set to 1.

On Nexus S we enabled the TCP window scaling to allow the burst sizes and burst intervals to vary. We used the traffic traces collected from our Linux server as input and calculated the aggregated energy cost with the threshold value of burst definition set to 5 ms. This threshold value was chosen as described in Section 4.3. We applied the model presented in (20) to estimate the power consumption from the burst information and the hardware parameter values listed in Table 6.

11. http://www.kismetwireless.net/

12. In the case of TCP download at 16KBps we observed from the traffic trace of N95 that there were two uplink bursts in a bin with the interval bigger than one PSM timeout. In the case of TCP upload at 16KBps on N95, the intervals between ACK and the following data packet was longer than the ACK timeout, which caused the retransmission of ACK. The retransmission wakes up the WNI and increases the duration spent in the IDLE mode. In both cases, the duration spent in the IDLE mode is longer than the PSM timeout, but no bigger than twice of the PSM timeout. For the sake of simplicity, we set the value of α to 2.

TABLE 7
Burst durations observed on N810, N95 and G1

Test case	Link	N810(ms)	N95(ms)	G1(ms)
TCP download	downlink	8	10	10
	uplink	0.5	0.35	0.5
TCP upload	uplink	6	12	8
	downlink	0.1	0.2	0.1

TABLE 8
MAPE of power models for N810, N95 and G1

Test case	PSM	N810	N95	G1
TCP download	On	4.3%±5.9%	4.9%±7.3%	-
(single flow)	Off	1.1%±0.7%	1.8%±2.9%	4.8%±2.7%
TCP upload	On	5.7%±4.9%	3.9%±5.5%	-
(single flow)	Off	2.2%±1.7%	2.3%±2.3%	1.3%±1.4%

We compared the estimated Power with the physical power measurement, and measured the estimation accuracy using MAPE as a metric. As listed in Table 8 and 9, the average MAPE of TCP download/upload with a single flow was at most 5.7%. We note that the results of Nexus S include the measurements in scenarios where the data is delivered with/without data rate limit.

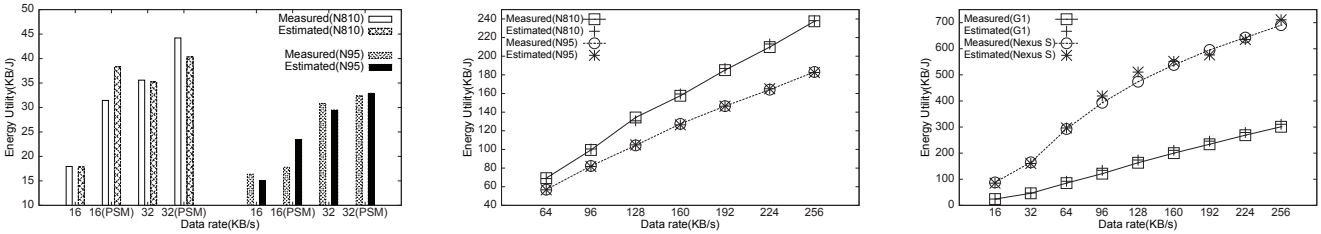
We also calculated the energy utility for each test case following (9). As shown in Fig. 5 and 6, the energy utility of TCP download/upload is nearly proportional to the network data rate. Thus it is more energy-efficient to transmit/receive data at a higher rate. In addition, when sending data at a rate lower than r_c, it is more energy-efficient if the PSM is enabled.

5.3 TCP Download/Upload with Multiple Flows

Since there can be one or multiple TCP flows transferring data from/to a mobile device, we extended our experiment from single TCP flow to multiple flows. According to the burst definitions in Section 3, packets belonging to different TCP flows are included in the same burst if they satisfy the two conditions of packet interval and transmission direction.

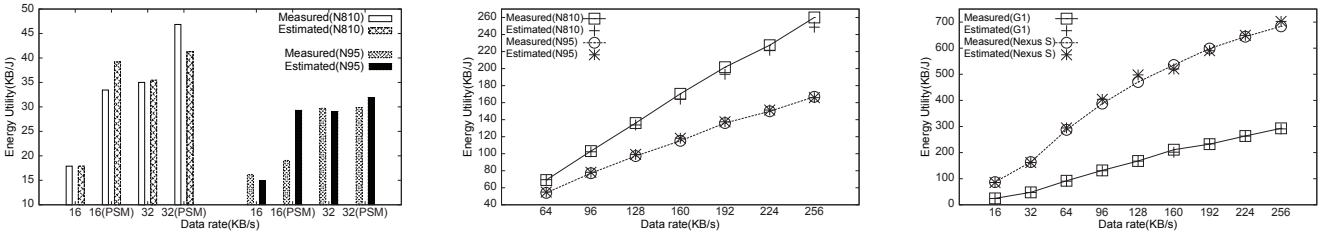
Assuming that there are N TCP flows running on one phone, we measured the power consumption of Nexus S in the following three scenarios: 1. All N flows downloading, 2. all N flows uploading, and 3. half of the flows downloading, other half uploading. For each scenario, we first ran the experiments without any data rate limit. After that, we used Trickle to set the limit of aggregate data rate to various values ranging from 512KBps to 2048KBps. The display was turned on with brightness set to 30% during the experiments. The experiments were repeated with N set to 2, 4 and 8, respectively.

We estimated the Power following (20), using the aggregated traffic information as input. The MAPE of our estimation was at most 9.1%, as shown in Table 9. We also calculated the mean and standard deviation of the energy utility at each data rate. Each data set corresponding to a data rate included all the results from the experiments using 2, 4 and 8 TCP flows respectively.



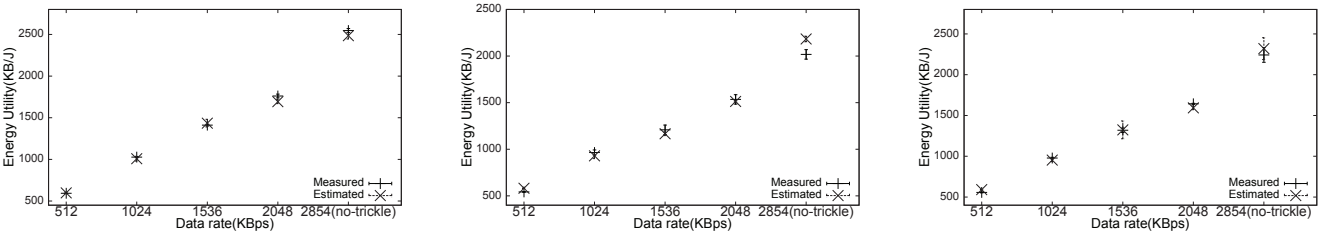
(a) On N810 and N95(16-32KBps). (b) On N810 and N95(64-256KBps). (c) On HTC G1 and Nexus S

Fig. 5. Energy utility of TCP download with single flow at rate between 16KBps and 256KBps.



(a) On N95 and N810(16-32KBps). (b) On N95 and N810(64-256KBps). (c) On HTC G1 and Nexus S.

Fig. 6. Energy utility of TCP upload with single flow at data rate from 16KBps to 256KBps.



(a) TCP download. (b) TCP upload. (c) TCP download and upload.

Fig. 7. Energy utility of TCP download/upload on Nexus S. The X-axis represents the aggregated data rate of all the TCP flows. The biggest value on the X-axis is the maximum throughput achieved without any rate limit. Each data point shows the mean and the standard deviation of the measured or the estimated Power.

TABLE 9
MAPE of power models for Nexus S

No. downlink	No. uplink	MAPE (%)	No. downlink	No. uplink	MAPE (%)
1	0	3.4±2.0⋆ 4.2±1.7	0	1	2.6±1.5⋆ 4.1±4.4
2	0	3.3±1.7⋆ 2.4±1.1	0 0	2 2	2.9±2.0⋆ 5.1±4.0
4	0	2.9±2.0	0	4	5.1±1.5
8	0	2.2±1.4	0	8	5.9±1.8
1	1	2.1±1.8⋆	1	1	2.7±1.9
2	2	5.0±2.8	4	4	4.8±2.1

⋆ The display was turned off. The data rate was between 16 and 256KBps.

As shown in Fig. 7, the standard deviation is very small compared to the value of the mean. As the processing overhead for maintaining more TCP flows is included in the measured Power, the small standard deviation of the measured Power shows that the processing overhead can be safely ignored. Fig. 7 also shows that the power models presented in Section 3 can provide generally accurate energy estimation of TCP transmission, regardless of the number of TCP flows.

5.4 TCP Download/Upload in Congested Network

We connected the Nexus S to a public AP in our campus and measured the power consumption during TCP download and upload. The phone tried to send/receive data as fast as possible without any data rate limit or traffic shaping. Due to the interference caused by the neighbouring APs, MAC layer retransmissions could not be left ignored. Based on the collected MAC layer traffic traces, we calculated the retransmission ratio R_r and the expected value of retransmitted packet interval $E(T_{ir})$. The samples of retransmitted packet intervals used for calculating $E(T_{ir})$ seem to follow the Inverse Gaussian distribution. The overhead of retransmitting packets was computed following (22). Because the CPU frequency was always 400MHz during the measurement, no extra cost was caused by DVFS. The final results are shown in Table 10. In upload cases, taking into account the retransmission overhead can improve the power estimation accuracy by almost 50%.

TABLE 11
Description of the experiments with Android apps

App	Description	Display	Throughput(KBps)	Duration(s)	App overhead(mW)⋆	MAPE
Fengxing video player	Stream videos from youku.com	on	7.9	3054	103	5.4%
Dropbox android app	Upload files to dropbox.com	on	18.9	1578	112	5.3%
Web browser	Open web pages on dalong.net	off	13.8	509	41	4.3%
QQ instant messenger	Receive random text messages	off	0.04	1068	127	8.6%

⋆ when the CPU is active

TABLE 10
MAPE of power estimation in congested networks

Test case	R_r(%)	$E(T_{ir})$(ms)	MAPE(%)
Download	11.2±1.5	-	2.4±1.2
Upload	20.6±2.3	0.8±0.2	5.2±4.1 (11.0±1.4⋆)

⋆ The retransmission overhead is not counted.

5.5 Real life applications

We evaluated the accuracy of the complete power models (Eq.(16), (17)), and the simplified one (Eq.(21)) with four real-life applications running on Nexus S. According to our measurements, the phone does not seem to implement the traditional 802.11 PSM/PSM adaptive with the WNI SLEEP mode, but does instead have a DVFS-induced low-power state that is entered upon expiration of an inactivity timer. This mechanism works in a manner similar enough to the 802.11 PSM that Eq. (17) can be applied with good results by replacing P_S with the measured power of the DVFS-induced low-power state, and the value of PSM timeout with the length of the inactivity timer.

The descriptions of the tested applications are listed in Table 11 along with experiment parameters and results. During the experiments, we ran Wireshark directly on the phone to capture the network layer traffic traces. The packet interval threshold used for burst detection was set to 5ms, and the overhead caused by the applications themselves[13] was included in the estimated values. We determined empirically the length of the inactivity timer to be about 1.35s when the display was off, and to be 400ms when the display was on. As shown in Fig. 8, all the models gave reasonable results, except in the case of the QQ messenger where only model (17) was able to estimate the power with good accuracy. This was caused by the exceptionally long burst intervals in the QQ messenger traffic, during which the DVFS put the phone into the low-power-state, which was only accounted for in model (17). Indeed, the traffic generated by the other 3 applications had at least 87% of the burst intervals shorter than 400ms. In the case of QQ messenger, only 21% of burst intervals were shorter than 400ms, while the WNI was estimated to stay in the SLEEP mode for more than 22% of the burst intervals.

5.6 Comparsion with PowerTutor

We chose the test case 3) defined in Table 3 as example and compared the accuracy of our approach with that of the readings from the PowerTutor [11] Android application. From Fig. 9 and 10 we can see that whereas the estimations of our model closely predict the measured values, the original PowerTutor readings deviate from the absolute measured values, and fail to follow the trends seen in the physical measurement. The high error rate can be explained by the fact the PowerTutor models had not been trained for the device in question (Nexus S), with the result that the value of the hardware-specific parameter $P_{baseline}$ might have been incorrect. Additionally, PowerTutor did not take into account the impact of CPU DVFS on the baseline power consumption. To reduce the bias caused by inaccurate $P_{baseline}$, we calibrated PowerTutor by configuring the value of $P_{baseline}$, which is hardcoded in the PowerTutor android application, with the values of P_I listed in Table 6. When the display was turned on, the PowerTutor android application estimated the power consumption of the display to be 183.6mW in our test cases. Because the power consumption of the display if available is already included in the value of P_I, we deducted 183.6mW from the calibrated results of PowerTutor in cases where the display was turned on.

Although the calibrated results of PowerTutor provided much better accuracy than the original ones, they still under-estimated the proportional effect of data rate on the power consumption. The MAPE of the calibrated PowerTutor readings is 20.74±12.96% in download cases and 12.93±10.86% in upload cases, which is higher than the MAPE of our models as listed in Table 9. This might indicate that the Wi-Fi power model of PowerTutor that according to [11] only takes into account the packet rate and the uplink channel rate as measured variables is too minimalistic to accurately describe the power consumption behaviour of real-life data transfer, where other variables such as traffic shape and CPU frequency also matter.

6 Discussion

The most recently commercialized standard 802.11n features several new features and enhancements compared to 802.11g. These features include MIMO support, channel bonding, and two power-saving mechanisms, namely Spatial Multiplexing Power Save (SMPS) and Power Save Multi-Poll (PSMP).

MIMO can be used either for spatial diversity by simultaneously transmitting redundant data streams encoded in a special way in order to increase range and robustness of data transmission or for spatial multiplexing

13. Measured Power with the app running without any data transfer.

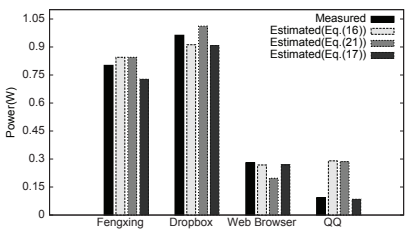


Fig. 8. Power consumption of 4 Android applications on Nexus S.

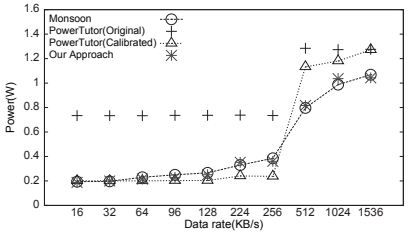


Fig. 9. Power consumption of TCP download with single flow on Nexus S.

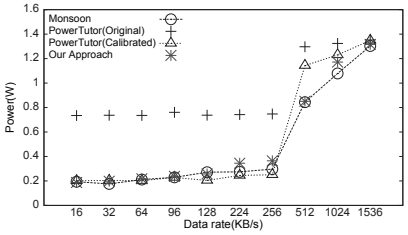


Fig. 10. Power consumption of TCP upload with single flow on Nexus S.

by transmitting multiple separate spatial data streams simultaneously in order to increase the transmission rate. In [26], the authors measured the energy consumption of 802.11n and discovered that only the number of active RF chains has a significant impact on the power draw and it does not matter much whether they are used for spatial diversity or multiplexing. Therefore, it is necessary to add the number of RF chains active at a given time instant as a parameter to capture MIMO characteristics in a power model.

Channel bonding allows to combine two adjacent 20 MHz channels into a single 40 MHz channel thereby doubling the bandwidth and transmission rate. Using a wider channel in this way has negligible impact on power consumption according to [26], which means that this feature can be also neglected by power models.

SMPS and PSMP do not alter the basic behaviour of the power saving mechanism of 802.11. Both reduce the energy consumed during idle periods. SMPS reduces the power draw when client is not receiving by switching off all but one RF chain. PSMP effectively makes it possible for the client to sleep as much of the idle time as possible. Considering our models, the impact of both of these mechanisms is reflected in the power levels measured for idle and sleep states and do not require specific modifications to the models.

The upcoming 802.11ac standard promises data rates beyond a gigabit per second which are achieved by taking advantage of more of the same above mentioned features of 802.11n. Specifically, 802.11ac can use wider channels (up to 160 MHz) and up to 8 spatial streams for MIMO operations. Thus, we have reason to expect that the same modelling approach will work for this family of products as well.

Our power models would serve well also in network simulators, such as NS-3. Integrating the models into such discrete event simulators is possible since the basic input to the models, namely packet headers, are readily available in such simulators. However, the support for accurate node models is presently limited [27]. For this reason, it is not possible to directly obtain DVFS related information of the simulated node. However, the different CPU frequency and voltage levels could be mapped to other measurable variables such as MAC level frame transmission/reception rate which the authors in [28] found to be highly accurate. Such mapping requires specific calibration for each different node device.

7 Conclusion

In this paper we present accurate and practical power models of Wi-Fi data transmission in infrastructure mode. To the best of our knowledge, our work is the first one that introduces traffic burstiness into power modeling of Wi-Fi data transmission. Our models quantify the impact from traffic patterns and network performance on the transmission cost, mainly caused by the operations of the WNI. These models can be used for estimating the power consumption of various network applications that are implemented with one to multiple TCP/UDP flows while being executed in varying network environments. We evaluate our models with physical power measurement of TCP-based data transmission on Maemo, Symbian and Android phones. The experimental results show that the MAPE of our power models is at most 9.1%, which is high enough for practical use. Furthermore, the characteristics of the transmission cost revealed in our power models do provide necessary motivation and insight into new solutions of energy-efficient Wi-Fi data transmission.

References

[1] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "Profiling resource usage for mobile applications: A cross-layer approach," in *Proc. of MobiSys '11*, June 2011, pp. 321–334.

[2] R. Friedman, A. Kogan, and K. Yevgeny, "On power and throughput tradeoffs of wifi and bluetooth in smartphones," in *Proc. of INFOCOM'11*, April 2011, 2011.

[3] M. Hoque, M. Siekkinen, and J. K. Nurminen, "On the energy efficiency of proxy-based traffic shaping for mobile audio streaming," in *Proc. of CCNC '11*, January 2011, pp. 891–895.

[4] A. Saarinen, M. Siekkinen, Y. Xiao, J. K. Nurminen, M. Kemppainen and P. Hui, "Can offloading save energy for popular apps?," in *Proc. of MobiArch '12*, August 2012, pp. 3–10.

[5] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *Proc. of MobiSys '10*, June 2010, pp. 49–62.

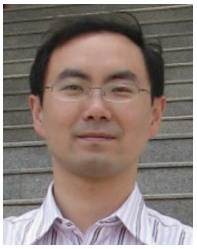
[6] E. Tan, L. Guo, S. Chen, and X. Zhang, "Psm-throttling: Minimizing energy consumption for bulk data communications in wlans," in *Proc. of ICNP'07*, October 2007, pp. 123–132.

[7] S. Mohapatra, N. Dutt, A. Nicolau, and N. Venkatasubramanian, "Dynamo: A cross-layer framework for end-to-end qos and energy optimization in mobile handheld devices," *IEEE J. Select. Areas Commun*, vol. 25, no. 4, pp. 722–737, May 2007.

[8] J. Garcia-Dorado, A. Finamore, M. Mellia, M. Meo, and M. Munafo, "Characterization of isp traffic: Trends, user habits, and access technology impact," *Network and Service Management, IEEE Transactions on*, vol. 9, no. 2, pp. 142 –155, June 2012.

[9] Y. Xiao, P. Savolainen, A. Karppanen, M. Siekkinen, and A. Ylä-Jääski, "Practical power modeling of data transmission over 802.11g for wireless applications," in *Proc. of e-Energy '10*, April 2010, pp. 75–84.
[10] J. Flinn and M. Satyanarayanan, "Managing battery lifetime with energy-aware adaptation," *ACM Trans. Comput. Syst.*, vol. 22, pp. 137–179, May 2004.
[11] 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 *Proc. of CODES/ISSS '10*, October 2010, pp. 105–114.
[12] M. Dong and L. Zhong, "Self-constructive high-rate system energy modeling for battery-powered mobile systems," in *Proc. of MobiSys '11*, June 2011, pp. 335–348.
[13] Y. Xiao, R. Bhaumik, Z. Yang, M. Siekkinen, P. Savolainen, and A. Ylä-Jääski, "A system-level model for runtime power estimation on mobile devices," in *Proc. of GreenCom '10*, Dec. 2010, pp. 27–34.
[14] A. Pathak, Y. C. Hu, M. Zhang, P. Bahl, and Y.-M. Wang, "Fine-grained power modeling for smartphones using system call tracing," in *Proc. of EuroSys '11*, April 2011, pp. 153–168.
[15] H. Singh, S. Saxena, and S. Singh, "Energy consumption of tcp in ad hoc networks," *Wirel. Netw.*, vol. 10, pp. 531–542, Sep. 2004.
[16] F. Qian, Z. M. Mao, Z. Wang, S. Sen, A. Gerber, and O. Spatscheck, "Characterizing radio resource allocation for 3g networks," in *Proc. of IMC '10*, Nov. 2010, pp. 137–150.
[17] 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 MobiSys '12*. June 2012, pp. 225–238.
[18] "IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*, pp. C1 –1184, June 2007.
[19] R. Fonseca, P. Dutta, P. Levis, and I. Stoica, "Quanto: Tracking energy in networked embedded systems," in *Proc. of OSDI'08*, Dec. 2008, pp. 323–338.
[20] B. Wang and S. Singh, "Analysis of tcp's computational energy cost for mobile computing," *SIGMETRICS Perform. Eval. Rev.*, vol. 31, pp. 296–297, June 2003.
[21] H. Jiang and C. Dovrolis, "Why is the internet traffic bursty in short time scales?" in *Proc. of SIGMETRICS '05*, June 2005, pp. 241–252.
[22] L. Guo, X. Ding, H. Wang, Q. Li, S. Chen, and X. Zhang, "Exploiting idle communication power to improve wireless network performance and energy efficiency," in *Proc. of INFOCOM'06*, April 2006, pp. 1–12.
[23] K.-c. Lan and J. Heidemann, "A measurement study of correlations of internet flow characteristics," *Comput. Netw.*, vol. 50, no. 1, pp. 46–62, 2006.
[24] M. A. Eriksen, "Trickle: A userland bandwidth shaper for unix-like systems," in *Proc. of USENIX ATC' 05*, April 2005, pp. 61–70.
[25] S. Agrawal and S. Singh, "An experimental study of tcp's energy consumption over a wireless link," in *Proc.of EPMCC'01*, 2001.
[26] D. Halperin, B. Greenstein, A. Sheth, and D. Wetherall, "Demystifying 802.11n power consumption," in *Proc. of HotPower'10*, Oct. 2010.
[27] S. Kristiansen, T. Plagemann, and V. Goebel, "Towards scalable and realistic node models for network simulators," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 418–419, Aug. 2011.
[28] A. Garcia-Saavedra, P. Serrano, A. Banchs, and G. Bianchi, "Energy consumption anatomy of 802.11 devices and its implication on modeling and design," in *Proc. of CoNEXT '12*, Dec. 2012, pp. 168–180.

Yu Xiao received the PhD degree in computer science from Aalto University in 2012. She is currently a postdoc researcher at the department of computer science and engineering, Aalto University. Her current research interests include energy-efficient wireless networking, crowd-sensing and mobile cloud computing.



Yong Cui Ph.D., Professor in Tsinghua University, Council Member in China Communication Standards Association, Co-Chair of IETF IPv6 Transition WG Softwire. Having published more than 100 papers in refereed journals and conferences, he is also the winner of Best Paper Award of ACM ICUIMC 2011 and WASA 2010. His major research interests include mobile wireless Internet and computer network architecture.

Petri Savolainen received his M.Sc from University of Helsinki in 2004. He is currently a post-graduate student at department of computer science, University of Helsinki and also a researcher at Helsinki Institute for Information Technology. His current research interests include peer-to-peer networking, energy-efficient computing, and mesh networking.

Matti Siekkinen obtained M.Sc. in computer science from Helsinki University of Technology and in Networks and Distributed Systems from University of Nice Sophia-Antipolis in 2003, and Ph.D from Eurecom / University of Nice Sophia-Antipolis in 2006. He is currently a senior research scientist at the department of computer science and engineering, Aalto University. His research interests include energy efficiency in ICT, network measurements, and Internet protocols.

An Wang received B.Sc in computer science from Beijing University of Posts and Telecommunications in 2010. He is currently pursing M.Sc. His research interests include energy-efficient wireless networking.

Liu Yang is an undergraduate student from Beijing JiaoTong University. Her research interests include energy-efficient wireless networking.

Antti Ylä-Jääski received his PhD in ETH Zuerich 1993. Antti has worked with Nokia 1994-2009 in several research and research management positions with focus on future Internet, mobile networks, applications, services and service architectures. He has been a professor for Telecommunications Software, Department of Computer Science and Engineering, Aalto University since 2004. His current research interests include green ICT, mobile computing, service and service architecture.

Sasu Tarkoma received his M.Sc. and Ph.D. degrees in Computer Science from the University of Helsinki, Department of Computer Science. He is full professor at University of Helsinki, Department of Computer Science and Head of the networking and services specialization line. He has over 100 scientific publications, and has authored three books. His interests include mobile computing, Internet technologies, and middleware.