PowerPi: Measuring and Modeling the Power Consumption of the Raspberry Pi

Fabian Kaup, Philip Gottschling, David Hausheer
Peer-to-Peer Systems Engineering Group, Technische Universität Darmstadt
Email: {fkaup|pgottschl|hausheer}@ps.tu-darmstadt.de

***Abstract*—An increasing number of households is connected to the Internet via DSL or cable, for which home gateways are required. The optimization of these – caused by their large number – is a promising area for energy efficiency improvements. Since no power models for home gateways are currently available, the optimization of their power state is not possible. This paper presents PowerPi, a power consumption model for the Raspberry Pi which is used as a substitute to conventional home gateways to derive the impact of typical hardware components on the energy consumption. The different power states of the platform are measured and a power model is derived, allowing to estimate the power consumption based on CPU and network utilization only. The proposed power model estimates the power consumption resulting in a RMSE of less than 3.3%, which is slightly larger than the maximum error of the measurements of 2.5%.**

I. Introduction

Considering the world's population of 7.1 billion and a fixed broadband subscription rate of 10.82% in 2012[1], there exist around 770 million home gateways world wide. With a power consumption of around 10 W each, their power draw alone results in approximately 6.7 TWh of electrical energy per year. This corresponds to 0.03% of the world electricity consumption[2], or between 2.6% and 5% of the Internet power consumption [19].

Green networks are an emerging topic. Baliga et al. estimate the carbon footprint of the Internet [1], which is extended by Hinton et al. to include the cost of content storage and delivery [11]. Chiaravigli et al. focus on Internet Service Provider (ISP) networks and the reduction of their power consumption [4]. Other, very active areas of energy improvements are cellular networks in general [9] and the optimization of 4G networks [5].

The development of energy models currently focuses mainly on servers [2], desktop PCs [6] or mobile handsets [25]. Less work is done on profiling the power consumption of home gateways, access points or small servers. Due to the high number of these devices [19], these cannot be neglected when evaluating the power consumption of the full Internet infrastructure.

43% of these devices located in the developed world are always on [8] and often idle. The idle resources of such devices may be used to provide network access to other users [21] or pre-load content for their owners, while they are away (i.e., caching, pre-fetching of content, running local services). The devices are built to be cheap and reasonably energy efficient. However, no detailed power model for this device class is available, which would allow for software based optimization. In most calculations [19] a fixed power consumption is assumed, independent of the device utilization.

The Raspberry Pi is a popular platform for low-power and low-cost computational tasks, suitable for a large range of applications. It is used as a platform to model cloud computing [22], for home monitoring and automation [18], or to provide low cost computation to developing countries [10]. Applications for the Raspberry Pi range from enhanced Internet gateways (Nano Datacenters (NaDas) [24], supporting the intelligent caching of video content, over intelligent WiFi access points [21], future ICN applications [16], to applications in outer space [3]. To accurately estimate the power consumption and possible improvements to each application, accurate power models of the devices are necessary.

To this end, this paper presents PowerPi, a power model focusing on the power consumption of the Raspberry Pi to derive possible power saving strategies. The hypotheses of the paper are:

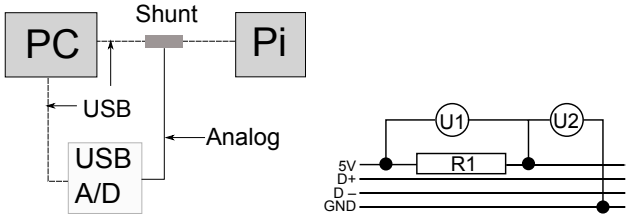
- An accurate estimation of the Raspberry Pi's power consumption can be obtained using the system utilization only.
- Knowing the power model, it is possible to reduce the power consumption by optimizing the software running on the platform.

PowerPi is based on hardware measurements of the Raspberry Pi.

The remainder of this paper is structured as follows. The setup and configuration of the platform is detailed in Section II, which gives an overview of tools and services used during the experiment and custom tools to generate load and monitor the system state. The measurements are described in Section III, showing a linear dependency between the CPU power consumption and the utilization. Similarly, second to fourth order functions between the data rate on the interface and the power consumption are derived. Based on these, the model generation and models approximating the behavior of the device under load are described in Section IV. The resulting model is compared to similar approaches in Section V. Section VI concludes the paper and gives an outlook on possible applications of the energy model.

[1]http://www.itu.int/net4/itu-d/icteye/, accessed 2013-12-04
[2]http://www.eia.gov/aeo, accessed 2013-12-04

978-1-4799-3780-6/14/$31.00 ©2014 IEEE



(a) Measurement setup (b) Wiring of the setup

Fig. 1. Measurement setup and its wiring for the Raspberry Pi

II. Measurement Setup

PowerPi measures the power consumption of the Raspberry Pi using an external power meter. Simultaneously, scripts and custom tools on the platform generate load and monitor the device state. The following sections describe the measurement setup in detail.

A. Power Measurement

The Raspberry Pi is a low-power device, which supports being powered via USB. Its power consumption is measured by interrupting the power lines of the USB connection and inserting a measurement shunt in the 5 V line. The wiring of the setup, as shown in Figure 1a, is detailed in Figure 1b. The current flowing through R_1 causes a voltage U_1, proportional to the current drawn by the Raspberry Pi. The requirements for the measurement shunt are twofold. First, it must be large enough to create a voltage that can easily be measured. Secondly, it must be small enough to reduce the voltage drop to a minimum, allowing the connected device to start. A resistance of 100 mΩ, with a maximum current of 1.2 A creates a voltage drop of 120 mV, which reduces the voltage on the +5 V line to 4.88 V. This is still sufficient to operate a USB device. Still, the voltage U_1 when only small currents are drawn is around 30 mV, allowing a sufficient accuracy. A 12 bit A/D converter with an absolute error of 6 mV results in a relative error of 20 %. Therefore, Measurement Computing's *USB1608-FSPlus* is used, which has a resolution of 16 bits, allowing the measurement of voltages of a few mV with an absolute accuracy of 0.68 mV, thus reducing the error to 2.3 % for idle measurements. The voltage U_2 between the 5 V line and *GND* is measured directly.

A custom built software based on Measurement Computing's *FlexDAQ API*[3] constantly measures the voltage drop U_1 over this shunt and the voltage U_2 of the 5 V line. The power consumption of the Raspberry Pi is calculated with

$$P_{\mathrm{Pi}} = \frac{U_1 \cdot U_2}{R_1}. \tag{1}$$

The software then writes the measurements together with a time-stamp to a local file.

The error of the power measurement depends on the accuracy of the two voltage measurements, which depend on the accuracy of the A/D conversion and the accuracy of the measurement shunt. Mathematically, the maximum error is defined as

$$\max\left(\frac{\Delta P}{P}\right) = \max\left(\sqrt{\left(\frac{\Delta I}{I}\right)^2 + \left(\frac{\Delta U}{U}\right)^2}\right) \tag{2}$$

The maximum error of the voltage measurement $\Delta U/U$ can be calculated directly from the above considerations as

$$\max\left(\frac{\Delta U_2}{U_2}\right) = \frac{\max(\Delta U_2)}{\min(U_2)} = \frac{5.66mV}{4.8V} = 0.12\% \tag{3}$$

Similarly, the maximum error of the current measurement is calculated.

$$\max\left(\frac{\Delta I}{I}\right) = \max\left(\sqrt{\left(\frac{\Delta U_1}{U_1}\right)^2 + \left(\frac{\Delta R_1}{R_1}\right)^2}\right) \tag{4}$$

The maximum error for the voltage U_1 is

$$\max\left(\frac{\Delta U_1}{U_1}\right) = \frac{\max(\Delta U_1)}{\min\left(U_1\right)} = \frac{0.68mV}{30mV} = 2.26\% \tag{5}$$

Combining the error of the measurement with the tolerance of the shunt of 1% results in an error of

$$\max\left(\frac{\Delta I}{I}\right) = \max(\sqrt{0.0226^2 + 0.01^2}) = 2.47\% \tag{6}$$

resulting in a maximum absolute measurement error of

$$\max\left(\frac{\Delta P}{P}\right) = \max(\sqrt{0.0247^2 + 0.0012^2}) = 2.47\% \tag{7}$$

This error is the upper bound of the errors introduced by the measurement setup. The actual accuracy of the measured samples is expected to be better. Furthermore, averaging the measurements over the evaluation period reduces the error of the final power measurements to even lower values.

B. Measurement PC Setup

These measurements are run from a conventional laptop. The only hardware requirements are a USB 2.0 interface for the measurement card and a Gigabit Ethernet interface to run the network tests. The Gigabit interface is recommended, as it ensures that the bottleneck of the throughput tests is located on the platform's network chip and not on the measurement PC. The Wireless Fidelity (WiFi) measurements can be run over a conventional WiFi Access Point (AP), or by creating a software AP on the laptop. Here, the over-provisioning of bandwidth on the remote side is difficult, as the WiFi interface selected supports the 802.11n standard. The measurements are best run from a Linux PC, as most software required for the measurements is readily available for this platform. Still, the power measurement software is written in Java, hence running measurements is possible on each OS.

Before running the tests, the measurement PC is prepared by installing and configuring a number of applications. A custom built software measures the voltages on the USB connection and writes the values to a CSV file. This is later evaluated and correlated with the measured system and

[3]http://www.mccdaq.com/daq-software/DAQFlex.aspx, accessed 2014-01-19

network utilization using MATLAB scripts. Furthermore, the Precision Time Protocol (PTP) [14] is run in server mode on the PC to allow the Raspberry Pi to synchronize its clock before executing measurements. *iPerf* is installed on both the measurement PC and the Raspberry Pi and run either in server or client mode to generate traffic in the respective direction. It is run as a daemon in UDP mode, as only this mode allows configuration of the target data rates. The system parameters such as current CPU utilization or consumed bandwidth are monitored on the Raspberry Pi itself.

C. Setup of the Raspberry Pi

The measured platform is a Raspberry Pi Model B running a software image based on Raspbian[4]. It is a Debian-based Linux distribution specifically designed for the Raspberry Pi. For the wireless tests, a USB WiFi dongle, the *D-Link DWL-G122* is used. It was selected because Linux drivers are available and high data rates (802.11n) are supported.

The default Raspbian image is extended by running PTP in client mode and a number of scripts monitoring the system utilization. The hardware monitors are detailed in Section II-D. Each monitor stores the collected measurements in the RAM until the end of the experiment. This minimizes the influence of the logging on the host system. The results are written to the SD-card only after the tests have finished.

D. System State Monitoring

During the tests only required services run, minimizing side effects. These services are *udev*, *dhcp client*, *ssh server*, and *dbus*. All other services are stopped after the boot process completes.

The system is monitored using custom scripts tracking state and utilization of the platform. As the CPU monitoring application also causes CPU utilization, special consideration was paid to reduce its influence to a minimum. Parsing the output of tools such as *top* or *ps* is resource intensive. A cause of this is the high amount of text output generated by these tools. Therefore, a very lightweight utilization monitor was written in *C* and compiled with the `-O3` option of *gcc* to optimize the generated code. To avoid disk I/O the monitors do not directly write to the memory card. The measurement script mounts a ramdisk to the `/tmp` folder and copies the content after the measurement is finished.

The CPU monitor reads the `/proc/stat` file (see Listing 1), which includes information about the number of cycles the CPU was busy (user, nice, system), in idle state, interrupted, and some more states since the last boot. The monitor takes the busy and idle states and computes the utilization according to Equation 9. Since the available values are incremental over time, the system utilization must be calculated from the difference of the counters. The system utilization $u[t]$ is calculated by dividing the busy cycles $c_{\text{busy}}[t]$ by the total number of cycles $c_{\text{total}}[t]$ during the last evaluation interval.

[4]http://www.raspbian.org/, accessed 2014-01-19

```
1 # comments do not belong to file
2 # cpu user nice system idle iowait irq softirq
3
4 cpu  13551 0 27452 665582 491 58 60 0 0 0
5 cpu0 13551 0 27452 665582 491 58 60 0 0 0
```

Listing 1. Excerpt of `/proc/stat`

The total number of busy cycles up to time t is defined as

$$c_{\text{busy}}[t] = c_{\text{user}}[t] + c_{\text{nice}}[t] + c_{\text{system}}[t]. \tag{8}$$

Here, $c_{\text{user}}[t]$ are the user generated CPU cycles, while $c_{\text{nice}}[t]$ and $c_{\text{system}}[t]$ are the cycles created by low priority processes and the system respectively. As we are interested in the full system load, these processes must be included in the calculation. The total number of cycles $c_{\text{total}}[t]$ is the number of busy cycles $c_{\text{busy}}[t]$ plus the number of idle cycles $c_{\text{idle}}[t]$ This leads to

$$u[t] = \frac{c_{\text{busy}}[t] \; - \; c_{\text{busy}}[t-1]}{c_{\text{total}}[t] \; - \; c_{\text{total}}[t-1]}. \tag{9}$$

Hence, the utilization at time t is calculated based on the difference of utilization cycles during the last measurement period.

Similar to the CPU monitor, a network monitor was written to keep track of the network utilization. The advantages of this approach are the low overhead, as the `proc` filesystem is used, the possibility to use any traffic generator and the elimination on parsing the output of bandwidth measurement tools. The drawback is the reduced accuracy of the first and last sample of an experiment. The influence of these is eliminated by running each test for a considerable time. The `/proc` filesystem is read at `/proc/net/dev`, returning the processed packets on kernel level. This is advantageous, as it contains the raw number of bytes sent and received via the interface. Similar to the `/proc/stat` file, the counters are incremental. The current bandwidth (in B/s) is calculated by

$$r[t] = \frac{B[t] \; - \; B[t-1]}{\Delta T} \tag{10}$$

where ΔT is the time interval between $t-1$ and t and $B[t]$ is the absolute amount of data transmitted or received on the interface.

E. Load Generation

To measure the different operating points of the CPU and the network interfaces, it is required to generate a configurable load. For this a combined approach of a load generator and load limiter is chosen. The CPU utilization is limited using a tool called *cpulimit*. It is available from the Raspbian repositories but has one major drawback for the measurements. It is designed to limit the CPU utilization of a specific process (including its children), but cannot control the overall CPU utilization. Hence, the *cpulimit* source code was modified to measure the full system utilization. The unmodified version of *cpulimit* reaches the targeted utilization very accurately because it has no external disturbance. After modifying the

```
1 int main(int argc, char* argv[]){
2         volatile int x=0;
3         volatile int y;
4         while(1){
5                 y=x+x;
6                 x++;
7         }
8         return 0;
9 }
```

Listing 2. Source Code to keep the CPU busy

source code, all other processes also influence the measured CPU utilization. Hence the variance is higher. The load is generated by running an infinite loop adding numbers as shown in Listing 2, filling up the CPU load to the desired limit.

The load on the network interfaces may be generated by a number of tools. The basic differentiation of these is between TCP and UDP connections. TCP is the most widely used protocol in the Internet, hence, its performance is of high interest. Still, UDP has a number of advantages considering the measurements. The most important aspect is the missing traffic control, which allows configuring a fixed data rate beforehand. This further eliminates the errors introduced by TCP's slow start and congestion avoidance algorithms. As UDP uses no back channel, the measurement of only the incoming or outgoing traffic counter is sufficient. Furthermore, only the bytes on the wire are used to calculate the power consumption. Considering the final model, the selection of UDP has no influence on the applicability of the final power model. Contrary, the accuracy of the measurements is improved. As the kernel file system is evaluated during the model generation, the power consumption generated by TCP traffic can be modeled as well.

As a traffic generator, *iPerf* was selected. It allows configuration of the bandwidth for UDP connections. The measurements are automated using a script running *iPerf* in different configurations. *iPerf* also returns traffic statistics during and after each run, but these miss the required accuracy. Furthermore, they only contain the self generated traffic and require parsing of the command line output.

III. Measurements

The measurements were conducted in a home environment during the night to reduce potential interference of the WiFi measurements. The power measurements are conducted with a sampling rate of 1 kS/s, while the maximum update rate of the bandwidth and utilization measurements is one sample per second. Hence, a block-wise average is applied to the power measurement samples. The start of the blocks was determined based on the beginning of the utilization samples. All measurement values between two system or network utilization samples are averaged and then mapped to the second value. The time difference between the utilization and bandwidth measurement points is always below 80 ms. Hence, the time difference may be neglected. Each test runs for 900 seconds,

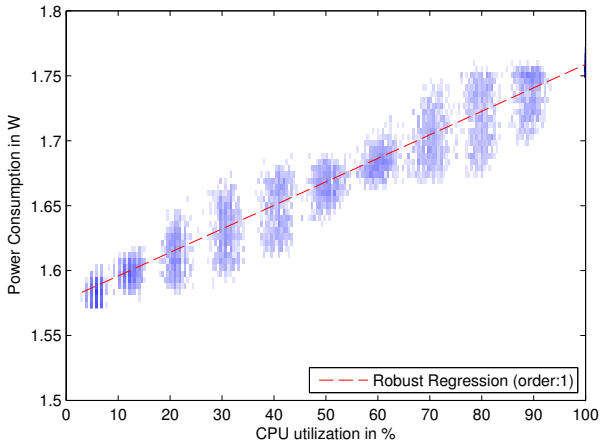


Fig. 2. Power consumption vs. CPU utilization

resulting in 900k power measurements, which are reduced to 900 combined utilization and power measurements. For each experiment 10 different operating points are configured and measured, resulting in 90k independent samples for each approximation. The CPU utilization measurement is executed without network access to reduce external influences to a minimum.

The collected measurements are plotted in a heat map to allow a visualization of the density of the measurements. This is advantageous compared to scatter plots, as the high number of measurements reduces the visibility of the individual data points. The heat map is logarithmically weighted to visualize the full range of measurements. On top of the heat map, the models derived in Section IV are plotted.

A. CPU

Figure 2 shows the result of the CPU measurements in the range 10% to 100% in 10% steps and a fitted linear function generated by Matlab's® `robustfit` function. The horizontal extent of the data spots reflects the accuracy of the CPU-limiting script. The variation is clearly smaller than ±5%. The power measurements between the different configurations are overlapping. This can be explained by the discrete number of frequency scaling steps in the processor. As the plotted values are averages of the individual measurements, the discrete steps are not visible. Still, this improves the visibility of the trend. As the number of power samples is 1000 times higher than the displayed values, the averages still reflect the underlying power consumption.

B. Ethernet

The power measurements in this section use the power model proposed in Section IV-A. From the raw measurements, the power consumption generated by the idle state and CPU is subtracted. Remaining is the power of the network transmission only. This is possible, as during the measurements, the network load and the CPU utilization were monitored by the respective scripts.

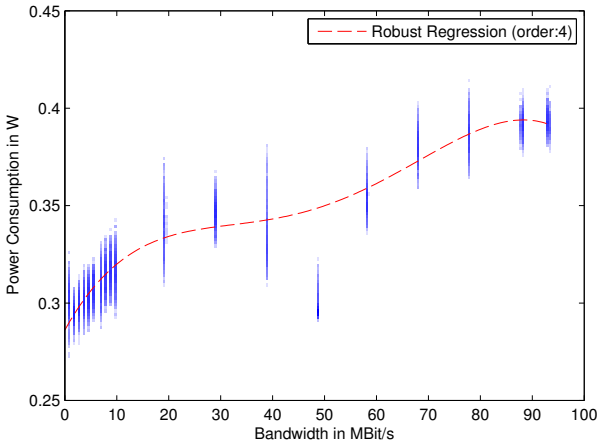


Fig. 3. Ethernet power consumption during download vs. used bandwidth

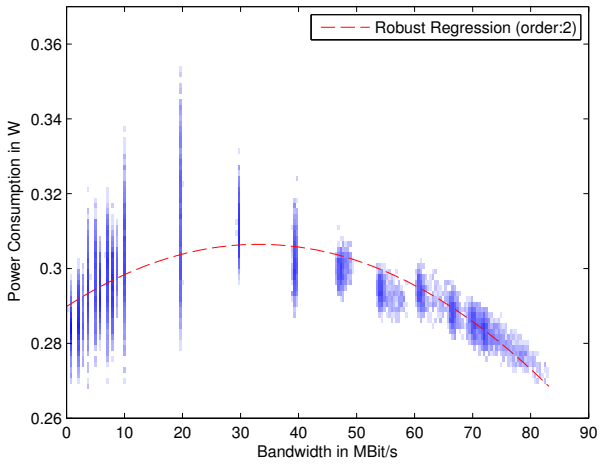


Fig. 4. Ethernet power consumption during upload vs. used bandwidth

Figure 3 shows the power measurements of the download experiments over the link utilization as a heat-map. The horizontal spread of the values denotes the variance in the measured throughput. The deviation is quite small for all data rates, indicating a low number of errors on the wire. Still, the data rate is slightly lower than requested. The power measurements stay in a range of 70 mW. There is one deviation in the measured data for a rate of 50 Mbps, where the power consumption is almost as low as the idle power. This measurement was repeated several times to rule out measurement errors. This deviation might be caused by the design of the hardware, which is in an optimal state for this rate. Still, the general trend of the power consumption is increasing while receiving data on the Ethernet interface, although the difference between the minimum and maximum is quite small.

Figure 4 shows the same configuration, but uploads from the platform are measured. The spread of the bandwidth measurements is quite narrow for lower traffic rates, which shows an accurate behavior of the traffic generator, while it is larger for higher rates. This effect is thought to be caused by the saturation of link and interfaces between the measurement PC and the Raspberry Pi. Higher data rates lead to higher collision probabilities, and hence, the actual data rates are lower than the requested ones. The variance of the power measurements is inverse to the variance of the traffic measurements. For low traffic rates, the variance is high, which is thought to be caused by frequent transitions between different power states within the device. For higher rates, where the link is saturated, this is less likely, which is also visible in the measurements above 50 Mbps. The maximum increase over the idle power for a fully utilized Ethernet interface while downloading is 96 mW at a rate between 30 Mbps and 40 Mbps. The lowest power consumption while uploading data is 21 mW lower than the power consumption of the idle interface. This behavior is also visible in other measurements of consumer grade network equipment [12], hence the result is plausible.

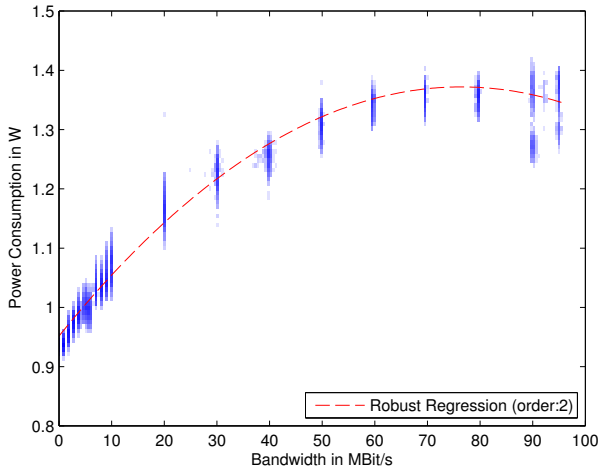


Fig. 5. WiFi power consumption during download vs. used bandwidth

C. WiFi

Figure 5 shows the power consumption when receiving data over WiFi. Contrary to the Ethernet measurements, a distinct increase in power consumption based over the down-link bandwidth is visible. The minimum power consumption of the interface is 950 mW while idle and increases to 1.4 W for a fully utilized link. Similar to the Ethernet down-link measurements, the variance of the measured data rates is quite low. The power measurements vary in a range of 100 mW, which is acceptable given the accuracy of the CPU measurements in Figure 2. Still, for higher data rates (>85 Mbps) two distinct power states are visible.

Figure 6 shows the power consumption of the WiFi interface during upload. Similar to the download tests, the spread of the measured data-rates is limited for lower bandwidths. For higher rates (>40 Mbps) the measurements begin to spread. This is thought to be caused by the higher computational effort required to create the frames and coordinate the connection.

This is also reflected in the power consumption, which is

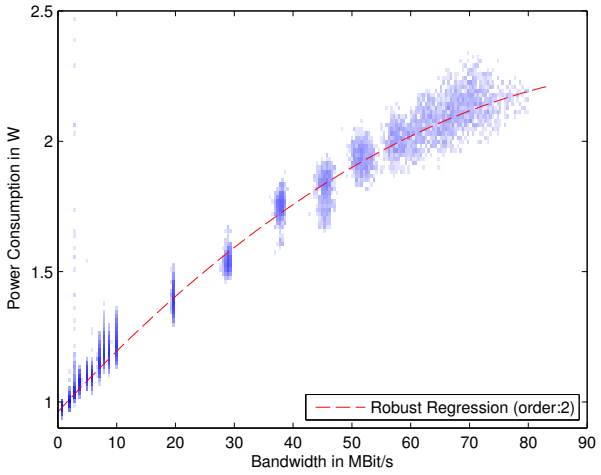


Fig. 6. WiFi power consumption during upload vs. used bandwidth

more than 700 mW higher at the upper end, when compared to the downloads. There is an interesting effect visible at the 3 Mbps measurement. This might be caused by the 802.11 protocol, switching to different coding schemes when interference is encountered. Still, this effect is not visible in the download measurements.

IV. Model Generation

The power model for the different measurements is generated by fitting a linear function to the measured data, minimizing the remaining root mean square error (RMSE). For this purpose, Matlab's™ `robustfit` function is used. This function is based on an iterative process fitting the linear function to the data, minimizing the RMSE. The detailed process is described in [13]. The underlying data are weighted with a bi-square function to reduce the effect of outliers on the final fit.

A. Description of the Power Models

The measurements of the CPU utilization in Figure 2 show a clear linear dependency. This observation is confirmed, by calculating the 1^{st} order regression. The resulting function for the platform including CPU utilization is

$$P_{\mathrm{Pi,CPU}}(u) = 1.5778\mathrm{W} + 0.181 \cdot u \cdot \mathrm{W}. \qquad (11)$$

Here, u is the CPU utilization in the range 0 to 1 as defined in Equation 9. The resulting RMSE is 18.9 mW, which corresponds to an error of 1.2%. The measurements below 10% CPU utilization represent the idle state of the Raspberry Pi without additional load.

Figure 3 shows the fourth order approximation to the Ethernet download measurements, for which Matlab's® `robustfit` function was used. The resulting RMSE of the 4th order function is 14 mW. This is only slightly lower than the RMSE of the first order approximation of 16 mW. For practical reasons, it might be sufficient to use the first order function when estimating the power. The equations for all approximations up to 4th order are detailed in Table I.

The graph in Figure 4 shows the second order approximation to the WiFi down-link measurements. The resulting RMSE of 8 mW shows the good fit of the model to the measured data. Still, for higher bandwidths, the error is increasing.

The WiFi download measurements in Figure 5 show a minimum (idle) power consumption of 950 mW, while the full utilization of the interface adds an additional 400 mW to this. The approximation plotted in the figure shows the second order model, resulting in a RMSE is 26 mW.

The power model plotted in Figure 6 is the second order approximation to the WiFi upload measurements. Looking at the data, also a first order model might be possible. The RMSE of the first order approximation is 83 mW, while the second order model results in a RMSE of 71 mW. Hence, the gain of using the higher order model is minimal.

B. Combined Power Model and Usage

Table I shows the approximations of the power consumption of the Raspberry Pi for different utilization. The first column is the symbol used in the text to refer to this function. The second column indicates the approximation order, while the third column gives the RMSE, which is the mean error to be expected when using the model. The last column lists the formula describing the dependency between utilization and power consumption.

The table is grouped to distinguish the different measurement categories. The first group gives the idle power consumption of the Raspberry Pi for different power states. $P_{\mathrm{Eth,idle}}$ and $P_{\mathrm{WiFi,idle}}$ denote the power draw when the platform is idle. The second group describes the power consumption of the platform depending on the CPU utilization. This is included as the variable u, giving the CPU utilization as defined in Equation 9. The remaining groups show the power consumption of the data transfers on both network interfaces. The first term of each model in these groups is a correction term necessary to fit the model to the constants determined before. The other terms are modeled to depend on the transferred data rate r in Mb/s or the CPU utilization u as defined in Equation 9. This results in an additive model, where the absolute power consumption of the Raspberry Pi can be modeled on a per-component basis. The model can be expressed as

$$P_{\mathrm{Pi}} = P_{\mathrm{idle}} + P_{\mathrm{CPU}}(u) + \sum_{\mathrm{if}} \left(P_{\mathrm{if,idle}} + P_{\mathrm{if,up}}(r) + P_{\mathrm{if,dn}}(r)\right), \qquad (12)$$

where the constants P_{idle} and $P_{\mathrm{if,idle}}$ and the approximations $P_{\mathrm{CPU}}(u)$ and $P_{\mathrm{if},d}(r)$ are defined in Table I. Here, the interface *if* is either *WiFi* or *Eth*.

The RMSEs of the built in components are generally quite low (<18 mW). Only the WiFi measurement shows a larger error. This is explained by a higher power consumption of the USB dongle, leading to a higher variance of the combined measurements. The power consumption of the USB WiFi dongle with 2 W is double the platform's power consumption, and close to the maximum allowed power draw of a USB 2.0 device of 2.5W.

TABLE I
THE POWER MODELS OF THE RASPBERRY PI. HERE, u IS THE CPU UTILIZATION IN THE RANGE 0 TO 1 AND r THE TRAFFIC RATE IN MB/S

Function	Ord.	RMSE	Model
P_{idle}	0	15 mW	1.5778 W
$P_{\mathrm{Eth,idle}}$	0	9 mW	0.294 W
$P_{\mathrm{WiFi,idle}}$	0	8 mW	0.942 W
$P_{\mathrm{CPU}}(u)$	1	18 mW	$0.181\mathrm{W} \cdot u$
$P_{\mathrm{Eth,dn}}(r)$	1	16 mW	$0.006\mathrm{W} + 1.060e^{-3} \cdot r \cdot \frac{\mathrm{W}}{\mathrm{Mbps}}$
$P_{\mathrm{Eth,dn}}(r)$	2	17 mW	$0.003\mathrm{W} + 1.634e^{-3} \cdot r \cdot \frac{\mathrm{W}}{\mathrm{Mbps}} - 6.531e^{-6} \cdot r^2 \cdot \frac{\mathrm{W}}{\mathrm{Mbps}^2}$
$P_{\mathrm{Eth,dn}}(r)$	3	16 mW	$-0.002\mathrm{W} + 2.702e^{-3} \cdot r \cdot \frac{\mathrm{W}}{\mathrm{Mbps}} - 3.838e^{-5} \cdot r^2 \cdot \frac{\mathrm{W}}{\mathrm{Mbps}^2} + 2.331e^{-7} \cdot r^3 \cdot \frac{\mathrm{W}}{\mathrm{Mbps}^3}$
$P_{\mathrm{Eth,dn}}(r)$	4	14 mW	$-0.008\mathrm{W} + 4.792e^{-3} \cdot r \cdot \frac{\mathrm{W}}{\mathrm{Mbps}} - 0.164e^{-3} \cdot r^2 \cdot \frac{\mathrm{W}}{\mathrm{Mbps}^2} + 2.509e^{-6} \cdot r^3 \cdot \frac{\mathrm{W}}{\mathrm{Mbps}^3 1} - 12.498e^{-9} \cdot r^4 \cdot \frac{\mathrm{W}}{\mathrm{Mbps}^4}$
$P_{\mathrm{eth,up}}(r)$	1	11 mW	$0.000\mathrm{W} + 2.327e^{-3} \cdot r \cdot \frac{\mathrm{W}}{\mathrm{Mbps}}$
$P_{\mathrm{eth,up}}(r)$	2	8 mW	$-0.002\mathrm{W} + 5.542e^{-3} \cdot r \cdot \frac{\mathrm{W}}{\mathrm{Mbps}} - 45.850e^{-6} \cdot r^2 \cdot \frac{\mathrm{W}}{\mathrm{Mbps}^2}$
$P_{\mathrm{WiFi,dn}}(r)$	1	59 mW	$0.057\mathrm{W} + 4.813e^{-3} \cdot r \cdot \frac{\mathrm{W}}{\mathrm{Mbps}}$
$P_{\mathrm{WiFi,dn}}(r)$	2	26 mW	$0.010\mathrm{W} + 11.003e^{-3} \cdot r \cdot \frac{\mathrm{W}}{\mathrm{Mbps}} - 71.988e^{-6} \cdot r^2 \cdot \frac{\mathrm{W}}{\mathrm{Mbps}^2}$
$P_{\mathrm{WiFi,up}}(r)$	1	83 mW	$0.064\mathrm{W} + 4.813e^{-3} \cdot r \cdot \frac{\mathrm{W}}{\mathrm{Mbps}}$
$P_{\mathrm{WiFi,up}}(r)$	2	71 mW	$0.020\mathrm{W} + 24.387e^{-3} \cdot r \cdot \frac{\mathrm{W}}{\mathrm{Mbps}} - 1128e^{-6} \cdot r^2 \cdot \frac{\mathrm{W}}{\mathrm{Mbps}^2}$

V. RELATED WORK

Currently, the main focus of related work is on analyzing and improving the energy efficiency of either high performance networking, mobile devices or network interfaces only. To the best of our knowledge, no power model for the Raspberry Pi is available, neither is there one for any of the other low-cost low-power ARM boards.

The power consumption of low power high efficiency processors (ARM, Intel ATOM) is compared to conventional high performance processors in [15]. Results of different benchmarks normalized by the power consumption of the processors are compared, concluding that, depending on the benchmark, the energy efficient variants of conventional desktop (i7) or server processors (Xeon E7) outperform ARM processors for most benchmarks in the number of achieved points per watt. The energy efficiency of ARM cores in a data-center environment is evaluated by Tudor et al. [23]. They analyze the combined impact of the CPU performance and the limitations of the memory access, which on ARM systems is generally slower compared to x86 CPUs, on the energy required to fulfill a particular tasks. As currently servers are usually operated in a CPU range of 10% to 50% [2], the power consumption while idle should also be included in the benchmarks.

Gomez et al. [7] have developed a power measurement and control board named *Energino*. They measure the power consumption of the WiFi adapter of a PCEngines ALIX 3D2 (500 MHz x86 CPU, 256 MB RAM) and generate an energy model for WiFi traffic. Measurements are taken using a custom built Arduino based power measurement platform (*Energino*) with a power resolution of 135 mW and time resolution of 10 ms. The absolute accuracy of the *Energino* is not detailed in the paper. Their strength is the low price of the measurement platform and the versatile placement options, which are achieved by including ZigBee communication chips for collecting the measurements. The measurements focus on deriving a model for traffic generated or received by the AP, but neglect the analysis of the computational complexity of WiFi encryption and the Ethernet interface.

Diouri et al. analyze the accuracy of different power metering approaches [6] for desktop and server machines. They compare the results of internal and external power measurements and analyze the influence of the sampling rate on the visibility of features on the generated graphs. Their conclusion is that the sampling rate must be adapted to the system being measured, but the accuracy of the sampling with different rates is not evaluated. Furthermore, averaging of measurements is not conducted to compare the overall accuracy over a given time interval.

Closest to this publication is the work by Nunez-Yanez et al. [17]. They analyze the processing and memory performance and energy consumption of an ARM Cortex-A9 chip manufactured for energy efficient processing in smartphones. Still, the influence of the network components is excluded.

VI. CONCLUSION AND OUTLOOK

This paper presented PowerPi, a power model for the Raspberry Pi, which includes the CPU and Ethernet power consumption as well as the power consumption of an external USB WiFi dongle. The power model is modular to incorporate all measured components.

At the beginning of the paper, the following hypotheses were made:

- An accurate estimation of the Raspberry Pi's power consumption can be obtained using the system utilization only.
- Knowing the power model, it is possible to reduce the power consumption by optimizing the software running on the platform.

The error of the model has been evaluated based on the measurement accuracy and the error introduced by the model. The resulting errors for the built-in components are in the order of tens of mW only. PowerPi can be used to improve the energy footprint of software running on the Raspberry Pi, using system traces only. Similar power models can easily be generated for other devices by repeating the same measurements. These power models are valuable, as future Internet services may likely run on the end-user's premises. Examples of possible future distributed services are NaDas [24], Multi-service Home Gateways [20] or HORST [21]. These aim at improving the Quality of Service (QoS) of the specific service and reducing the dependency on the up-link bandwidth, while requiring local computation and storage.

Using the PowerPi model, and similar models of other devices connected to the network, it becomes possible to estimate the power consumption of the full network for a given load. Hence, energy efficiency improvements of the full network infrastructure are possible based on the power models and the system utilization only. This eliminates the need for dedicated power measurement hardware, but allows derivation of the power consumption with an accuracy of lower than 3.3%. For a networked system specialized on traffic forwarding, the traffic statistics alone are sufficient to generate accurate predictions of the network state and power consumption. This allows optimizing the traffic flows to use the most energy-efficient paths, or redirect computations to the most energy-efficient location based on the current utilization. Thus, both hypotheses are supported by this paper.

ACKNOWLEDGMENTS

This work has been supported in parts by the European Union (FP7/#317846, SmartenIT and FP7/#318398, eCousin). The authors would like to acknowledge valuable comments by their colleagues and project partners.

REFERENCES

[1] J. Baliga, K. Hinton, R. Ayre, and R. S. Tucker, "Carbon footprint of the Internet," *Telecommunications Journal of Australia*, vol. 59, no. 1, pp. 1–14, 2009.

[2] L. A. Barroso and U. Hölzle, "The Case for Energy-Proportional Computing," *Computer*, vol. 40, pp. 33–37, 2007.

[3] J. Berk, J. Straub, and D. Whalen, "The Open Prototype for Educational NanoSats: Fixing the Other Side of the Small Satellite Cost Equation," in *IEEE Aerospace Conference*, 2013.

[4] L. Chiaraviglio, M. Mellia, and F. Neri, "Minimizing ISP Network Energy Cost: Formulation and Solutions," *IEEE/ACM Transactions on Networking*, vol. 20, no. 2, pp. 463–476, 2012.

[5] P. Dini, M. Miozzo, N. Bui, and N. Baldo, "A Model to Analyze the Energy Savings of Base Station Sleep Mode in LTE HetNets," in *IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, 2013, pp. 1375–1380.

[6] M. E. M. Diouri, M. F. Dolz, O. Glück, L. Lefèvre, P. Alonso, S. Catalán, R. Mayo, and E. S. Quintana-Ortí, "Solving Some Mysteries in Power Monitoring of Servers: Take Care of Your Wattmeters!" in *Energy Efficiency in Large Scale Distributed Systems*, ser. Lecture Notes in Computer Science, J.-M. Pierson, G. Da Costa, and L. Dittmann, Eds. Springer Berlin Heidelberg, 2013, vol. 8046, pp. 3–18.

[7] K. Gomez, R. Riggio, T. Rasheed, D. Miorandi, and F. Granelli, "Energino: a Hardware and Software Solution for Energy Consumption Monitoring," in *International Workshop on Wireless Network Measurements*, 2012, pp. 311–317.

[8] S. Grover, M. S. Park, S. Sundaresan, S. Burnett, H. Kim, and N. Feamster, "Peeking Behind the NAT: An Empirical Study of Home Networks," in *IMC*, 2013.

[9] Z. Hasan, H. Boostanimehr, and V. K. Bharghava, "Green Cellular Networks: A Survey, Some Research Issues and Challenges," *IEEE Communications Surveys and Tutorials*, vol. 13, no. 4, pp. 524–540, 2011.

[10] R. Heeks and A. Robinson, "Ultra-low-cost computing and developing countries," *Communications of the ACM*, vol. 56, no. 8, pp. 22–24, Aug. 2013.

[11] K. Hinton, J. Baliga, M. Feng, R. Ayre, and R. S. Tucker, "Power Consumption and Energy Efficiency in the Internet," *IEEE Network*, no. April, pp. 6–12, 2011.

[12] H. Hlavacs, G. Da Costa, and J.-M. Pierson, "Energy Consumption of Residential and Professional Switches," in *International Conference on Computational Science and Engineering*, 2009, pp. 240–246.

[13] P. W. Holland and R. E. Welsch, "Robust regression using iteratively reweighted least-squares," *Communications in Statistics - Theory and Methods*, pp. 813–827, 1977.

[14] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–269, 2008.

[15] M. Jarus, S. Varette, A. Oleksiak, and P. Bouvry, "Performance Evaluation and Energy Efficiency of High-Density HPC Platforms Based on Intel, AMD and ARM Processors," in *Energy Efficiency in Large Scale Distributed Systems*, ser. Lecture Notes in Computer Science, J.-M. Pierson, G. Da Costa, and L. Dittmann, Eds. Springer Berlin Heidelberg, 2013, vol. 8046, pp. 182–200.

[16] U. Lee, I. Rimac, D. Kilper, and V. Hilt, "Toward Energy-Efficient Content Dissemination," *IEEE Network*, vol. 25, no. 2, pp. 14–19, 2011.

[17] J. Nunez-Yanez and G. Lore, "Enabling Accurate Modeling of Power and Energy Consumption in an ARM-based System-on-Chip," *Microprocessors and Microsystems*, vol. 37, no. 3, pp. 319–332, May 2013.

[18] K. Okiah, C. Finn, P. Leal, and T. Mirabito, "Power Pi - The Green Revolution," University of Massachusetts Amherst, Tech. Rep., 2012.

[19] B. Raghavan and J. Ma, "The Energy and Emergy of the Internet," in *HotNets*, 2011.

[20] Y. Royon and S. Frénot, "Multiservice Home Gateways: Business Model, Execution Environment, Management Infrastructure," *IEEE Communications Magazine*, vol. 45, pp. 122–128, 2007.

[21] M. Seufert, V. Burger, and T. Hoß feld, "HORST - Home Router Sharing based on Trust," in *SETM Workshop*, 2013, pp. 402–405.

[22] F. Tso, D. T. White, S. Jouet, J. Singer, and D. Pezaros, "The Glasgow Raspberry Pi Cloud: A Scale Model for Cloud Computing Infrastructures," in *International Workshop on Resource Management of Cloud Computing*, 2013.

[23] B. M. Tudor and Y. M. Teo, "On understanding the energy consumption of ARM-based multicore servers," in *International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*. New York, New York, USA: ACM Press, 2013, p. 267.

[24] V. Valancius, N. Laoutaris, L. Massoulié, C. Diot, and P. Rodriguez, "Greening the Internet with Nano Data Centers," in *CoNEXT*, 2009, pp. 37–48.

[25] L. Zhang, B. Tiwana, R. P. Dick, Z. Qian, Z. Mao, Z. Wang, and L. Yang, "Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones," in *CODES + ISSS*. ACM, 2010.