Northumbria
University
NEWCASTLE

A Real-time Power Monitoring and Energy-efficient Network/Interface Selection Tool for Android Smartphones

Mehmet Fatih Tuysuz[1], Murat Ucan[1], Ramona Trestian[2]
[1] Department of Computer Engineering, Harran University, Şanlıurfa, Turkey
[2] School of Science and Technology, Middlesex University, London, UK

***Abstract*—Energy efficiency in wireless and cellular networks has become one of the most important concerns for both academia and industry due to battery dependence of mobile devices. In this regard, Wireless Network Interface Cards (WNICs) of mobile devices have to be taken into account carefully as they consume an important chunk of the system's total energy. In this paper[1], we propose a real-time network power consumption profiler and an energy-aware network/interface selection tool for Android-based smartphones. The tool has been freely released on the Android Play Store. The proposed solution reports the power consumption levels of different network interfaces (Wi-Fi and Cellular) by making use of actual packet measurements and precise computations, and enables the devices to handover horizontally/vertically in order to improve the energy efficiency. In this context, widespread analyses have been executed to show the accuracy of the proposed tool. The results demonstrate that the proposed tool is very accurate for any type of IEEE 802.11 wireless or cellular stations, regardless of having different amount of channel utilization, transmission rates, signal strengths or traffic types.**

Keywords*: Energy-efficient Network/interface Selection, Android, Wireless & Cellular Networks*

I. INTRODUCTION

The high affordability of smart high-end mobile devices along with the growing popularity of rich multimedia-based applications, led to a tremendous growth in mobile data traffic that puts significant pressure on the underlying network technology. Cisco is predicting that by 2021 [1], the traffic from wireless and mobile devices will account for more than 63% of the total IP traffic. In this context, no single network technology will be equipped to deal with this explosion of data, making the fifth generation (5G) wireless communications seen as a promising solution for heterogeneity and interoperability as illustrated in Figure 1. Thus, an important and key enabling technology for 5G is the heterogeneous network (HetNet) which should integrate the coexistence of different radio access technologies (RATs) to enable the provisioning of Quality of Service (QoS) and Quality of Experience (QoE) to the mobile users.

Even though the advancement of computational capability and processing power at the mobile device enables the mobile users for rich multimedia-based experiences on the go, the excessive battery dependency bounds the run-time of these devices. Additionally, despite the fact that processing power doubles almost every two years, the progress in batteries has not even doubled over the last decade [2]. Works in [3-7] depicts that excessive energy volume, up to 10–50%, can be consumed through wireless/cellular interfaces (such as, Wi-Fi, GSM, 3G and LTE) among all energy-hungry components (e.g., screen, processors, sensors, communication and computation circuitries) of a smartphone. For this reason, companies and researchers have begun to modify the design of protocols and network interface cards for mobile devices, taking energy efficiency into consideration.

In addition, since mobile devices equipped with heterogeneous wireless interfaces continually seek channels to initiate handovers, proposing an energy-efficient horizontal/vertical handover method is also substantial to diminish power consumption while still providing necessary QoS. In this context, the handover process and its correctness are vital for the

[1] This research has been funded by the Scientific and Technological Research Council of Turkey (TÜBİTAK) with Grant No: 114E075.

energy efficiency, since a probable incorrect connection to a new network may result in devices to consume even more energy than before until a suitable connection, if ever, is chosen [3].

Regarding the energy-efficient interface/network selection, we first suggested a computational method in [8] that lets devices connect to a Point of Attachment (PoA) that is projected to consume the least amount of energy among all PoAs in the vicinity, taking critical factors such as RSS, channel density and WNIC power into account. Moreover, in [9], we extended our former work and proposed an energy-aware network/interface selection and handover application for Android-based mobile devices. To the best of our knowledge, it was the first application in Google Play Store that enables devices to handover both horizontally and vertically. The application presented in [9] briefly computes/reports power consumption levels of each PoA in the vicinity for various web-applications (e.g. Facebook, Twitter and Skype), utilizing actual packet measurements and precise computations, and then lets the devices handover horizontally/vertically to improve their energy efficiency.

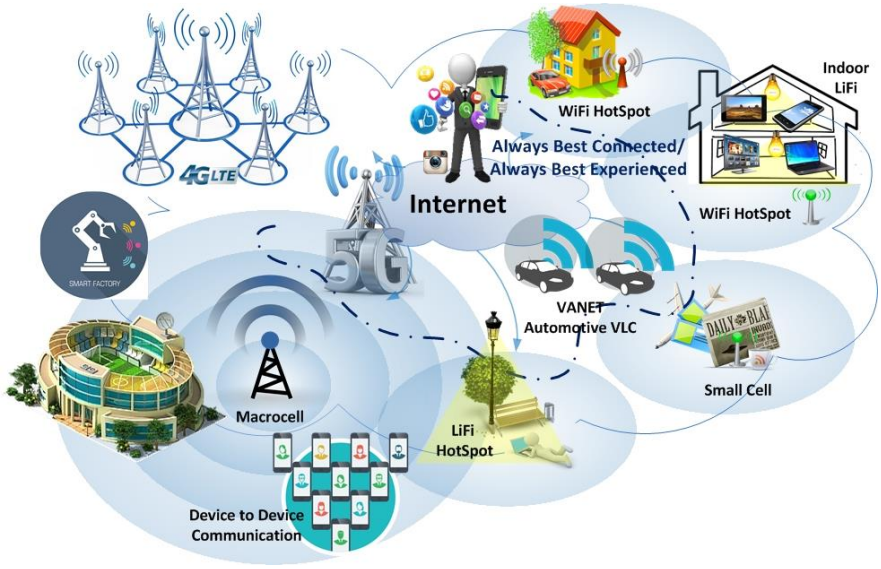


Figure 1. Heterogeneous Wireless Environment – Example Scenario

In this context, this paper builds on the previous work presented in [9] and proposes a real-time network power consumption profiler and an energy-aware network/interface selection tool for Android-based mobile devices. The proposed solution, overcomes the shortcomings of the previous work by providing accurate results for any type of wireless or cellular mobile devices, regardless of the varying channel utilization, transmission rates, signal strengths or traffic types.

The main contributions of this paper are as follows:

- The proposed solution integrates a PowerProfiler tool with an energy-aware network/interface selection tool. The proposed network PowerProfiler reports power consumption levels of wireless/cellular interfaces in real-time by utilizing actual packet measurements and precise computations. Within the scope of this work, an in-depth analysis of the existing power profilers and their faulty/missing parts are explained at first. Then, performance of the proposed work is examined under different scenarios in the evaluation section, such as additional energy cost that is consumed by the application, impact of channel utilization, signal strength and RATs on power consumption

and throughput. Whereas, the work in [9] can support energy-aware vertical handover but not real-time power monitoring.

- In contrast to the previous studies [8, 9], the LTE interface, in addition to Wi-Fi and 3G, has also been studied and included into the computations of the proposed solution. In this way, the power monitoring and energy-efficient handover operations are managed more accurately with the addition of LTE interface.
- Within the scope of this study, user preferences were added to the application for user-centric increase of energy efficiency. Users can now manage various operations such as auto-handover, movement activation, repetition-based scanning, add already known APs, and set threshold values for power-related parameters and step counter.
- Android service class and motion detection approach running in background have also been updated within this study so that the application can perform its operations consuming less energy.

The remainder of the paper is structured as follows: Section II reports existing network power monitoring tools and energy-aware network selection approaches proposed in the literature. Section III presents an in-depth analysis of the existing power profilers and their faulty/missing parts. Section IV introduces the power consumption estimation of the wireless/cellular interfaces. Section V describes the concept of the proposed real-time network power consumption profiler and energy-efficient network/interface selection tool. Section VI presents the evaluation process, with the focus on energy efficiency achieved by the approaches proposed in Section V under different scenarios. Finally, Section VII summarizes the implications of this work.

II. RELATED WORKS

There have been numerous Android-based mobile tools in the literature that recommend devices, which network to connect to, such as: *WiFi Analyzer Lite, WiEye - WiFi Scanner, WIFI Scan, Wifi Scanner, Wi-Fi Analytics Tool, etc.* Nevertheless, these tools focus only on the Wi-Fi interface and so the vertical handover is not addressed. Moreover, these tools consider only the Received Signal Strength Indicator (RSSI) as the main decision criteria.

Current energy-aware network/interface selection schemes [10-24] are either mobile-initiated or network-assisted and generally run depending on a specific metric like cost-function [10, 12], traffic-volume [13, 15], fuzzy-logic [14,22], mobility pattern-aware [16], location-assisted [17], context-aware [18], speed-sensitive [20], delay-tolerant [23], etc. For instance, authors in [25] propose a fuzzy vertical handoff algorithm that triggers handoff from a resource to another, when the energy consumption of the device increases or the connection time with the resource decreases. Authors in [26] focus on resolving sticky client problem (i.e. sticking with the same interface even if signal quality is severely degraded) through fast WiFi handoff. In this context, they analyze the causes of sticky client problem based on experiments with commercial Android smartphones by focusing on WiFi scanning and handoff operations. Authors in [27] propose a multi-RAT interface activation algorithm with supporting system design for smartphones' file transfer service. The goal is to find out the optimal multi-RAT set to be activated with the corresponding file segment allocation that minimizes the cost function under given energy and quota constraints. In this context, authors model a multiattribute cost function incorporating file-transfer completion time, energy consumption, and data usage quota together. Furthermore, authors in [28] propose MAPS, low-power AP monitoring scheme for handover decision triggering in heterogeneous networks. Using MAPS, a mobile device can trigger a handover decision properly through predicting the connected AP's network condition accurately without any

cooperation from other devices. Although above-mentioned works are designed to save energy and increase overall throughput, they have three well-known issues; (i) computation of power consumption with insufficient knowledge, (ii) using quite the same energy projection for different RATs, and (iii) acquiring results under specific channel/traffic condition.

In order to meet the requirements of saving energy, it is also essential to observe and examine power consumption levels of running tools and network interfaces on the device. There are several Android OS based mobile tools [29-35] that monitor the power consumption levels of running applications and network interfaces. Some of the most well-known tools are PowerTutor [29], DevScope [30], AppScope [31], Semo [32], Intel Performance Viewer [33], Trepn Profiler [34] and GSam Battery Monitor [35]. Most of these tools mainly collect information about foreground applications and display it in a real-time graph. They are also able to list/rank applications according to the amount of power they consume on various components, such as Wi-Fi, 3G, CPU, screen, etc. Yet, our detailed wireless/cellular power consumption analysis and computations have revealed that all of the abovementioned tools actually fail at measuring/estimating the actual network power consumption level. In order to compensate the faulty and missing parts, an in-depth analysis of the well-known profilers is presented in the following sections. We also show why assumptions/computations of existing network power profiling tools are faulty or missing. Finally, to address this issue, we propose a new real-time network power consumption profiler tool for Android smartphones.

III. ANALYSIS OF THE EXISTING POWER PROFILERS

This section provides an in-depth analysis of the proposed WiFi and 3G power models for wireless and cellular networks.

A. Analysis of the Existing Power Profilers in terms of WiFi Power Consumption

PowerTutor[2] [29] is one of the most well-known power profilers and it is also the only profiler that can calculate the amount of power consumption of a specific application on-line. It visualizes the obtained information through pie charts, line graphs and ordered lists. PowerTutor also enables generating log files for a more detailed analysis. In PowerTutor, Wi-Fi power consumption relies on several parameters, such as: (i) number of packets transmitted/received, (ii) uplink channel rate and (iii) uplink data rate. PowerTutor assumes that Wi-Fi interface has four power states: (i) low_power, (ii) high_power, (iii) l_transmit and (iv) h_transmit. l_transmit and h_transmit are the states the network card enters when transmitting data. At the end of transmission, the network card returns to its previous state. PowerTutor assumes wireless stations transit from low_power to high_power when more than 15 packets are transmitted or received per second. If the number of transmitted/received packets drops to 8 or less, then the station returns to the low_power states.

In a similar manner, Wi-Fi power states are categorized in four states by DevScope [30]: (i) L_idle, (ii) H_idle, (iii) L_transmission, and (iv) H_transmission. In DevScope, outgoing packets are only available in L_transmission and H_transmission states. Additionally, Wi-Fi power pattern is separated by thresholds, so that if the packet rate is less than 20 packets/sec., Wi-Fi is in the L_transmission state, otherwise it is in the H_transmission state.

[2] PowerTutor sets power consumption in l_transmit and h_transmit states to 1000mW. In low_power state, it is set to 20mW and in high_power state, it is set to 710mW.

In AppScope [31], the amount of power consumption is set according to the transmitted packets per second as in the DevScope. The number of transmitted packets is calculated utilizing the transmission/reception interval of each packet. The power state is then identified based on the packet rate, and power consumption is calculated utilizing activated time durations. SEMO [32] does not perform an on-line Wi-Fi power consumption estimation. It only ranks running applications by the expected power consumption rates. Ranking applications according to their expected power consumption is simply achieved by periodic records of time, remaining battery lifetime and the applications running at that time.

Intel Performance Viewer [33] is able to measure how many Kbytes are being sent and received per application. However, it is not able to differentiate between Wi-Fi and 3G technologies. Similarly, GSam Battery Monitor [34] can also measure the amount of data that is sent or received over Wi-Fi and 3G, but does not differentiate the traffic between these two RATs. Trepn Profiler [35] is able to differentiate Wi-Fi and 3G, and it can analyze the amount of data that is sent or received. However, none of the abovementioned three tools is able to compute the amount of power consumed by the Wi-Fi interface.

It should be noted that all of the abovementioned tools actually fail at measuring/estimating the actual Wi-Fi power consumption level. First of all, uplink and downlink traffic might have different transmission rates and sizes, therefore must be analyzed separately. In fact, each transmitted/received frame must be analyzed separately as they may also have different transmission rates and sizes. Moreover, the power states durations (*transmitting*, *receiving*, *idle* and *doze*) actually vary according to the number of stations (n_s), probability of collision (p_c) and Frame Error Rate (FER). Therefore, in order to compute/estimate how much a station is expected to stay in each of the four states accurately, it is also essential to integrate the channel density and signal quality into the equation.

As an instance, Figure 2 illustrates the variation of *transmitting* and *idle* state durations of a device that aims to transmit five frames based on different RSSI, channel-busy-time (CBT) and traffic types (e.g. TCP or VoIP). In the figure, black and stripped pulses characterize successful and unsuccessful transmission attempts, respectively. In *Case 1*, high CBT and low RSSI result in a collision and a frame error, so that the expected time intervals that the device will stay in the *transmitting* and *idle* state increase ($t_t = t_{t1} + t_{t2} + t_{t3} + t_{t4} + t_{t5} + t_{error} + t_{collision}$, $t_{idle} = t_{idle1} + t_{idle2} + t_{idle3} + t_{idle4} + t_{idle5} + t_{idle6}$). In contrast, low CBT and high RSSI in *Case 3* enable the device to transmit only five frames without any error or a collision. Therefore, the station consumes less power in *Case 3* compared to *Case 1*.

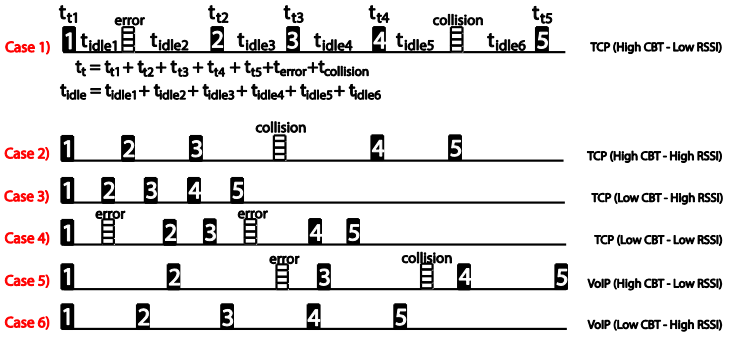


Figure 2. Variation of transmitting and idle state intervals of a device.

Accordingly, as is shown in Figure 2, a change in any of the indicators also changes the *idle* and *transmitting* intervals of a device and so, the total power consumption. Hence, all of the indicators must be cautiously examined to achieve precise power consumption ratios of different scenarios. Last but not the least, the amount of power consumption also depends on the WNIC chipset and the wireless standard (each of IEEE 802.11a\b\g\n\ac has different amount of power consumption level) running on the device.

In a nutshell, all of the above-mentioned facts prove that existing Android OS based Power Consumption Profilers provide inaccurate Wi-Fi power consumption results. Therefore, a new algorithm that accurately reports power consumption levels of Wi-Fi interface, utilizing actual packet measurements and precise computations, is required.

B. Analysis of the Existing Power Profilers in terms of Cellular Power Consumption

It should be noted that all of the existing profilers reviewed above do not provide a power monitoring tool for the LTE interface. Therefore, the reviewed models will be evaluated only for the 3G interface.

In PowerTutor, the 3G power model depends on transmitting and receiving rates (*data rate*) and three power states: Cell_DCH, Cell_FACH and IDLE. Power consumption is set 570mW, 401mW and 10mW for the Cell_DCH, cell_FACH and IDLE states, respectively. In case there is no packet transmission/reception activity for a fixed time (inactivity timer 2), 3G interface leaves the Cell_DCH state and enters the Cell_FACH state. In case the queue size exceeds its threshold, Cell_DCH state is activated again. In contrast, if the interface is idle for a fixed time (inactivity timer 1), the IDLE state is initialized. In the IDLE state, only paging messages are received. In case of any packet transmission/reception activity, 3G interface is transferred to the Cell_DCH state again. In DevScope and AppScope, power consumption of 3G interface only depends on the power state transitions. In a similar manner to the PowerTutor, 3G interface has three types of RRC states in both DevScope and AppScope, which are DCH, FACH, and IDLE. Other existing applications (SEMO, Intel Performance Viewer, Trepn Profiler and GSam Battery Monitor), on the other hand, are not able to compute the amount of power consumed by the 3G interface.

It should be noted that, for a precise power consumption calculation, it is significant to clarify all the dynamics that have an effect on power consumption. Total power consumption of transmitting/receiving packets includes roughly the *ramp energy* (the energy consumed during the state transition from the idle to DCH state), the *transfer energy* (the energy consumed during the transfer of the data chunk through the radio interface) and the *tail energy* (the energy consumed during the intermediate power states after the transmission) [36]. The *transfer energy* can be negligible in case of transferring/receiving only a single or few packets. Nevertheless, considering a session of continuous packet transmission, *transfer energy* consumes the highest portion, and importance of the *ramp* and *tail energy* would be much lower (insignificant) since sequential packet transmission keeps the medium in transmitting state uninterruptedly.

IV. ESTIMATION OF THE AMOUNT OF NETWORK POWER CONSUMPTION

A. Estimation of the amount of power consumed by WiFi Interface

The amount of expected power consumption of a Wi-Fi interface can be calculated roughly as follows,

$$P_{expected} = P_T(i,j) + P_R(i,j) + P_I(i,j) \quad (1)$$

where $P_T(i,j)$, $P_R(i,j)$ and $P_I(i,j)$ are the amount of expected power consumption of the network *i* and the traffic type *j* in the *transmitting*, *receiving* and *idle* state, respectively. Accordingly, each of these parameters can be calculated depending on the state intervals as follows,

$$P_T(i,j) \cong P_{T\_NIC} \times T_{Transmitting} \quad (2)$$

$$P_R(i,j) \cong P_{R\_NIC} \times T_{Receiving} \quad (3)$$

$$P_I(i,j) \cong P_{I\_NIC} \times T_{Idle} \quad (4)$$

where P_{T_NIC}, P_{R_NIC} and P_{I_NIC} are the power consumption of the WiFi interface card in the *transmitting*, *receiving* and *idle* state, respectively. Here, $T_{Transmitting}$ relies on the number of packets that will be transmitted, data rate, frame size, frame error rate (FER) and collision probability P_c. Accordingly, $T_{Transmitting}$ can be calculated as follow,

$$T_{Transmitting} \cong \frac{s(j)}{R_{up}} \times N_f \times (1 + FER) \times (1 + P_c) \quad (5)$$

Here, $s(j)$ is the size of transmitted packets, R_{up} is the packet transmission rate, N_f is the number of packets transmitted. Finally, $(1 + FER) \times (1 + P_c)$ is equal to the packet loss rate. Accordingly, expected amount of power consumption of a device in the *transmitting* state is roughly equal to,

$$P_T(i,j) \times T_{Transmitting} \cong P_{T\_NIC} \times \frac{s(j)}{R_{up}} \times N_f \times (1 + FER) \times (1 + P_c) \quad (6)$$

In a similar way, $T_{Receiving}$ is roughly equal to,

$$T_{Receiving} \cong \frac{s(j)}{R_{down}} \times N_f \times (1 + FER) \times (1 + P_c) + \frac{s(j)}{R_{ACK}} \times N_{f\_ack} \times (1 + FER) \times (1 + P_c) \quad (7)$$

Consequently, expected amount of power consumption of a device in the *receiving* state is roughly equal to,

$$P_R(i,j) \times T_{Receiving} \cong P_{R\_NIC} \times [\frac{s(j)}{R_{down}} \times N_f \times (1 + FER) \times (1 + P_c) + \frac{s(j)}{R_{ACK}} \times N_{f\_ack} \times (1 + FER) \times (1 + P_c)] \quad (8)$$

where R_{down} is the downlink transmission rate and P_{R_NIC} is the power consumption of the network interface card in the *receiving* state. Finally, T_{Idle} depends on the observation time t_{obs} (i.e. time interval between the first time the environment is started to be monitored and the time monitoring is terminated) and the time spent in the *transmitting* and *receiving* states, respectively.

$$T_{Idle} \cong t_{obs} - T_{Transmitting} - T_{Receiving} \quad (9)$$

Hence, expected amount of power consumption of a device in the *idle* state is roughly equal to,

$$P_{idle}(i,j) \times T_{Idle} \cong P_{idle\_NIC} \times (t_{obs} - T_{Transmitting} - T_{Receiving}) \quad (10)$$

B. Estimation of the amount of power consumed by 3G Interface

Power consumption of a 3G interface depends on the three power state transitions; Cell_DCH, Cell_FACH and Cell_IDLE. In the Cell_DCH state, stations consume power both to maintain the Cell_DCH state, and to send or receive frames. Therefore, the amount of expected power consumption of a 3G interface can be roughly computed as follows,

$$P_{expected} \cong (P_{DCH} + P_{tr/rcv}) \times T_{DCH} + P_{FACH} \times T_{FACH} + P_{IDLE} \times T_{IDLE} \quad (11)$$

where P_{DCH} is the power consumption of maintaining the Cell_DCH state, $P_{tr/rcv}$ is the power consumed in transmitting/receiving packets. P_{FACH} and P_{IDLE} are the power consumption of maintaining the Cell_FACH and Cell_IDLE states, respectively. Finally, T_{DCH}, T_{FACH} and T_{IDLE} are the durations of 3G interface in DCH, FACH and IDLE states. Our proposed algorithm computes the expected power consumption of 3G interface through power state transitions and transferring/receiving packets (n Mbps data) per second. The number of transfer blocks (N_{tb}) required for transferring/receiving n Mbps data can be computed as the ratio of n to the Maximum Transport Block Size (MTBS).

$$N_{tb} = \frac{n}{MTBS} \quad (12)$$

Since the energy cost of transferring/receiving n Mbps data also depends on the packetization interval (I_p) of frames and the successful transfer rate ($S(C_x)$) in the Point of Attachment (PoA_x), the amount of expected power consumption of a 3G interface that is connected to PoA_x can be re-computed as follows,

$$P_{expected} = P_{DCH} \times \left(\frac{N_{tb} \times I_p}{S(C_x)} + T_{a\_it1}\right) + P_{tr/rcv} \times \left(\frac{N_{tb}}{S(C_x)}\right) + P_{FACH} \times (T_{FACH} + T_{a\_it2}) + P_{IDLE} \times T_{IDLE} \quad (13)$$

where T_{a_it1} and T_{a_it2} are additional time intervals that the station stays in the Cell_DCH and Cell_FACH states, respectively due to inactivity timers. In Cell_FACH state, data rate is no more than a few hundred bytes per second. Switching between Cell_FACH and Cell_DCH states are carried out according to the downlink/uplink queue sizes (Table 2) in the RNC. Whenever the predefined queue size exceeds its threshold, Cell_DCH state is entered. In contrast, in the case where the 3G interface of the station is idle for a predefined interval, the IDLE state is entered.

C. Estimation of the amount of power consumed by LTE Interface

The power consumption of an LTE interface depends on two basic power state transitions: CONNECTED and IDLE. High power is consumed in the CONNECTED state as the radio is always on, and either transmitting or receiving data takes place. On the other hand, low power is consumed in the IDLE state as the radio is off, and no transmission/reception occurs. The LTE radio states consume a bit more power than 3G states, due to tail states in LTE stay at the higher base power, while much of the 3G tail is in the Cell_FACH state, which uses half power [37].

LTE power consumption model in [38] has been integrated in this study due to the model being proved to be successful and the difference between the measured values and the proposed model in [38] is only between 2% and 4.1% despite various investigating scenarios being carried out. Based on the analysis of the four physical layer parts (receiving (Rx) and transmitting (Tx) power levels, uplink (UL) and downlink (DL) data rates), the power consumption model from [38] is defined as follows:

$$P_{extepted} \cong m_{idle} \cdot P_{idle} + m_{con.} \cdot \{P_{con.} + m_{Tx} \cdot m_{Rx} \cdot P_{Rx+Tx} + m_{Rx} \cdot [P_{Rx} + P_{RxRF}(S_{Rx}) + P_{Rx} + P_{RxBB}(R_{Rx}) + m_{2CW} \cdot P_{2CW}] + m_{Tx} \cdot [P_{Tx} + P_{TxRF}(S_{Tx}) + P_{TxBB}(R_{Tx})]\} \quad (14)$$

where $P_{extepted}$ is the expected power consumption. P_{idle} and $P_{con.}$ are the power consumption values in RRC idle and connected mode, P_{RxRF} and P_{TxRF} are power consumption values of the RF part in Rx and Tx chains. P_{RxBB} and P_{TxBB} are power consumption values of the baseband parts, and 2CW is related to increased consumption when using two code words (CW) in download. The parameters P_{Rx}, P_{Tx}, and P_{Rx+Tx} are included to model the base power the Rx and Tx chains consume when active. In addition, the variable m represents the mode, which can be RRC idle, transmitting, receiving, and indicates the use of 2 CW. The Rx and Tx power levels are designated by S. Finally, R represents the Rx or Tx data rate.

V. CONCEPT OF THE PROPOSED POWERPROFILER AND ENERGY-AWARE NETWORK SELECTION

A. Network PowerProfiler

The flowchart of the PowerProfiler application developed within the scope of this study is given in Figure 3. The application starts with the calculation of how many packets are transmitted or received within the analyzed time period. In the proposed application, the number of packets transmitted ($currentTransmittedPackets$) and received ($currentReceivedPackets$) are calculated at every 10 milliseconds (msec.) intervals. Similarly, how many bytes are transmitted and received within the same time slot is also calculated.

$$CurrentByte_{Transmitted} = LastByte_{Transmitted} - PreviousByte_{Transmitted.} \quad (15)$$

$$CurrentByte_{Received} = LastByte_{Received} - PreviousByte_{Received.} \quad (16)$$

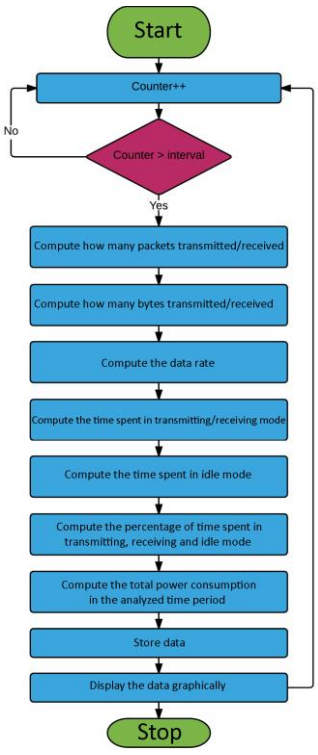


Figure 3. Flowchart of the proposed Network PowerProfiler application.

It should be noted that the $currentTransmittedPackets$ and $currentReceivedPackets$ data are obtained from the /proc directory of the mobile device. The proc file system on Unix systems is mounted in the /proc directory. This file contains information about the status of various files in the system. That is, you can open files and read from them to get information about the kernel. Many Unix programs (e.g. *gdb*, *ps*, *top*, and *truss*) use the /proc file system to control processes and collect information about them.

According to the transmitted/received packets, transmission rate ($\frac{s(j)}{R_{up,down}}$) is also calculated by the application for both uplink and downlink traffic. Next, time intervals the device stays in the *transmitting* and *receiving* modes are calculated per interface at 10-msec. intervals. Then, the time elapsed in the *idle* mode is calculated per time period as follows,

$$T_{idle} = 10msec. - T_{receiving} - T_{transmitting} \quad (17)$$

Finally, total power consumed in the analyzed time period is calculated per interface, the data is recorded and graphically displayed by the application.

B. Energy-aware Network Selection

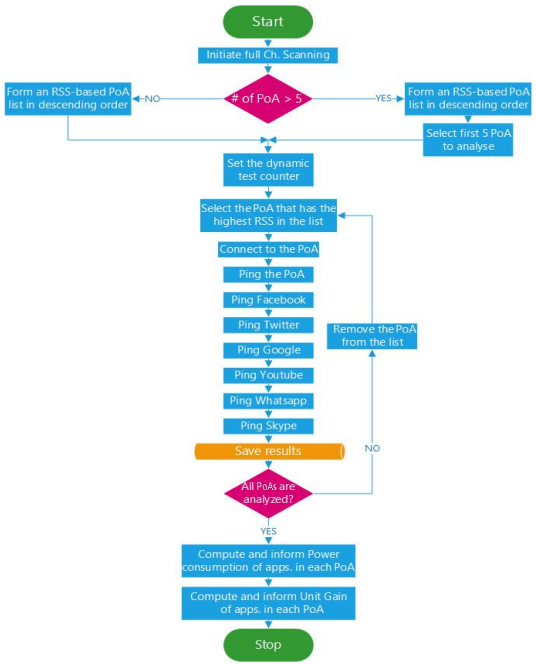


Figure 4. Flowchart of the proposed energy-aware PoA selection algorithm.

Figure 4 depicts the flowchart of the proposed energy-efficient network/interface selection algorithm. The algorithm first enables devices to perform full passive channel scanning. If the number of PoAs discovered is more than five, it creates an RSS-based PoA list in descending order and selects only the first five PoA that have higher RSS than the others. Afterward, the device running the algorithm connects to the PoA that has the highest RSS in the list. The device transmits ping messages to the PoA, and saves the average RTT interval, average RSS and also the packet loss rate (PLR) for that PoA. The device then determines the IP addresses of the servers of popular web-applications (Facebook, Twitter, Google,

YouTube, WhatsApp and Skype) by *nslookup host_url* command. Whenever the device acquires these IP addresses, it transmits ping messages to the IP addresses of these web-applications in an order, and then saves the average RTT intervals, average RSSs and also the PLRs for all of these web-applications. At the end of this process, the algorithm removes the currently-analyzed PoA from the list and repeats the same procedure until there is no PoA to be examined.

Figure 5. Multi-threaded flowchart of the proposed ping transmissions when there are two available APs.

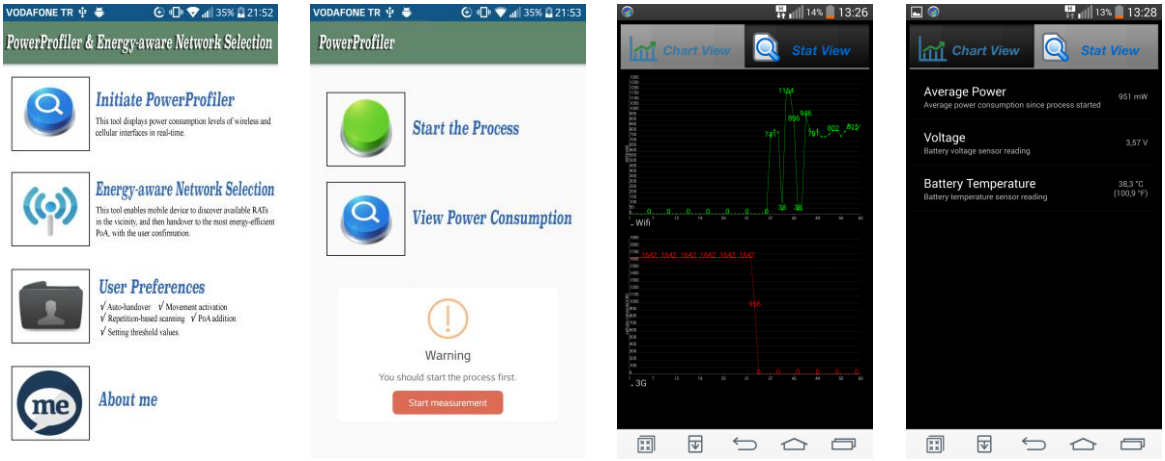
In order to minimize the processing time of the algorithm, multi-threaded programming was also applied. In this context, *AsyncTask* is used to manage waiting processes of threads that work simultaneously, and require results of an earlier thread(s) to terminate. *AsyncTask* class enables processes that run on different threads in the background to integrate into the main thread that run on foreground with ease. Thus, drawback of multi-threaded structure, in terms of processes waiting for one another, is also handled, utilizing the *AsyncTask*. As an instance, multi-threaded flowchart of the proposed energy-efficient network selection tool in case there are only two available PoAs is shown in Figure 5.

The proposed tool makes use of ping messages transmitted consecutively with 1-msec. intervals. To be able to transmit pings with 1-msec. intervals, root permissions have to be granted by the device. The command that allows devices to

transmit ping messages with 1 msec. intervals is; "*ping –c ping_count –i 0.001 host*", where *-c* is used to set how many pings (*ping_count*) will be sent, *-i* is used to set the transmission interval in between two consecutive pings. The value 0.001 means the interval is set as 1 msec. Finally, the *host* is the IP address or the URL that pings will be transmitted to.

C. Overview of the Proposed Application

Figure 6-a shows the home screen of the proposed tool that can be downloaded from Android Google Play Store[3]. The application basically has two closely related tools; (i) power consumption monitoring and (ii) energy-efficient network/interface selection. The proposed Network PowerProfiler reports power consumption levels of wireless/cellular interfaces by utilizing actual packet measurements and precise computations. Through the home screen of the PowerProfiler (Figure 6-b), it is possible to perform a test measurement by transmitting/receiving fixed number of packets to/from the server installed, using the "*Start the Process"* button, or continuously monitor real-time power consumption values of Wi-Fi, 3G and LTE interfaces. As shown in Figure 6-c, PowerProfiler displays the estimated amount of power consumed by the network interface of mobile devices in a chart view. The application also reads sensor data to display the remaining battery voltage and battery temperature as shown in Figure 6-d.



a) Home screen b) PowerProfiler home screen c) Chart view d) Statistical view

Figure 6. Screenshots of the PowerProfiler tool

In addition, the proposed tool also enables devices to find available RATs and their channel frequencies in vicinity, order them according to the expected amount of energy usage, and then handover to the most energy-efficient PoA, with the user confirmation. In this regard, screenshot of the proposed tool in process is shown in Figure 7-a. The tool performs channel scanning and orders the available networks based on the expected amount of power consumption. Within the scope of this study, user preferences (Figure 7-b) were also added to the application for user-centric increase of energy efficiency. Users can now manage various operations such as auto-handover[4], movement activation[5], repetition-based scanning[6], add already known Wi-Fi[7] (Figure 7-c), and set threshold values[8] for power-related parameters (Figure 7-d).

---

[3] https://play.google.com/store/apps/details?id=com.muratucan.PowerProfiler

[4] The device will automatically handover in case a more energy-efficient PoA is detected.

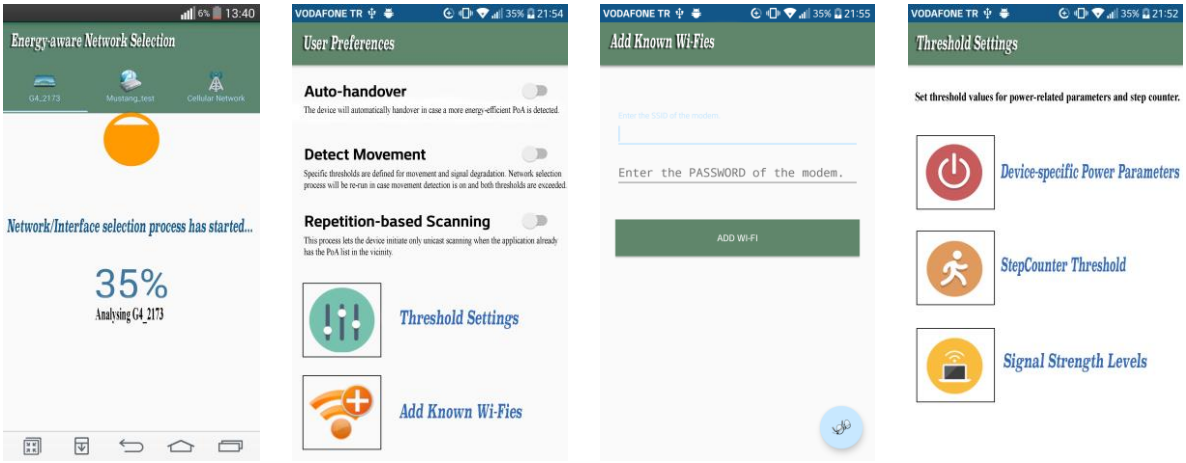
[5] This process defines specific thresholds for movement and signal degradation, and re-initiates the application if both thresholds are exceeded.

[6] This process lets the device initiate only unicast channel scanning when the application already has the PoA list in the vicinity.

[7] Proposed application let users type/store SSIDes and passwords of already known Wi-Fies beforehand. This way, in case auto-handover option is also activated, application will automatically handover whenever a more energy-efficient network/interface is found.

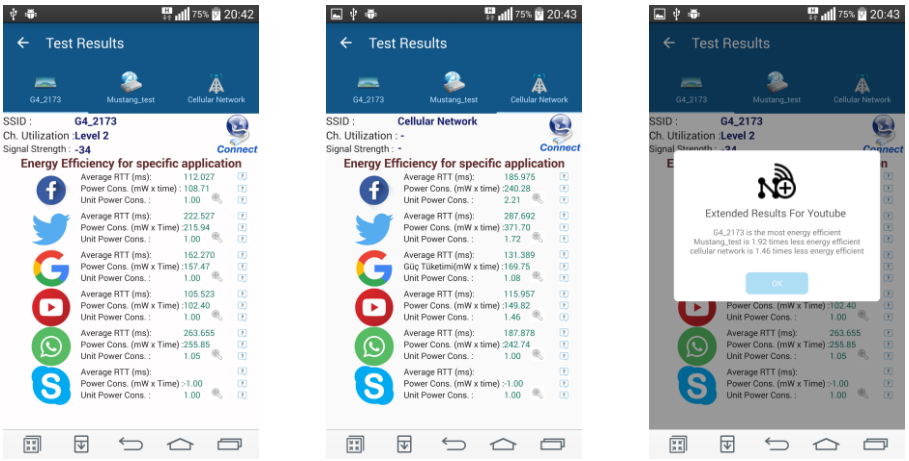
[8] Application enables threshold settings for device-specific power parameters, step counter and signal strength levels.

Test results of a sample set-up that has two Wi-Fi and one cellular network, obtained by running the proposed tool, are shown in Figure 8. While results of specific networks are shown in single pages (Figure 8-a, and Figure 8-b), results of a specific web-application, such as YouTube, are shown in detail in a different page once clicked (Figure 8-c).



(a) Application in process b) User preferences c) Adding known Wi-Fies d) Threshold settings

Figure 7. Energy-aware network selection and handover tool.



(a) Results of a Wi-Fi network (b) Results of Cellular Network (c) YouTube results

Figure 8. Test results of a sample set-up.

VI. EVALUATIONS

Within this section, four different scenarios have been tested. Within the scope of this study, all the tests were run 10-times and the average results were plotted. Measured results and the results produced by the application have been discussed in detail at the end of each sub-section. Examined scenarios are briefly related to: (i) the impact of additional energy cost that is consumed by the application, (ii) impact of handover on throughput and power consumption, (iii) impact of AP channel utilization on throughput and power consumption, and (iv) impact of different signal strength levels and RATs on throughput and power consumption. The parameters used by the proposed application to compute the expected amount of power consumption of the device that is connected to either a WLAN, a 3G or a LTE network are shown in Table 1, Table 2 and Table 3[9], respectively.

[9] Parameters in Table 1, Table 2 and Table 3 are derived from the works in [35-41].

Name	Value	Unit
Transmitting state	1.3	W
Idle state	0.74	W
Receiving state	0.9	W
Data rate	11	Mbps
DIFS	50	µs
SIFS	10	µs
Slot time	20	µs

Table 1. Parameter values of IEEE 802.11b WLAN.

Name	Value	Unit
P_{tr}	2.8	W
P_{IDLE}	0.82	W
P_{FACH}	0.42	W
P_{rcv}	0.49	W
P_{DCH}	0.85	W
Data rate	7.2	Mbps
Inactivity timer_1	3	Second
Inactivity timer_2	3	Second
Downlink/uplink queue size	130	Bytes

Table 2. Parameter values of 3G network.

Name	Value	Unit
P_{idle}	0.42	W
P_{con}	1.51	W
P_{Rx}	0.38	W
P_{Tx}	0.51	W
P_{Rx+Tx}	0,19	W
P_{2CW}	0.07	W
Data rate	14.4	Mbps

Table 3. Parameter values of LTE network.

AppScope and DevScope do not allow instantaneous/continuous power consumption to be viewed over smartphones since these applications estimate power consumption level of a device only after the device is connected to a computer by a cable. Therefore, the proposed application is only compared with the PowerTutor[10] throughout this section.

A. Additional Energy Cost that is Consumed by the Application

Additional energy will be consumed by the device running the proposed application based on the execution of different processes that perform necessary actions, such as computing the total number of transmitted/received packets per second, detecting any movement, counting steps, analyzing/finalizing the results, and plotting the results on the screen. Consequently, these processes start consuming energy whenever initiated, and energy consumption continues as long as the application is in use. Therefore, additional energy cost that is consumed by the application must be analyzed.

In order to reveal the energy necessity of additional processes, we have executed measurements for four specific states; (i) total power consumption measured after a 100-minute test period when the proposed application is off, (ii) total power consumption measured after a 100-minute test period when the proposed application is on, but the *Movement Detection* is off, (iii) total power consumption measured after a 100-minute test period when both the proposed application and the *Movement Detection* is on and the smartphone is stationary, and (iv) total power consumption measured after a 100-minute test period when the proposed application is on, the *Movement Detection* is on, and the smartphone is frequently on move[11].

[10] Although PowerTutor and PowerProfiler have different parameter values, same parameter values depicted in Table 1 and Table 2 were used for the two applications in this study, so that the applications can be compared with each other.

[11] During the tests, we frequently walked around the computer networks lab while having the device in our hand, and staying inside the coverage of the connected AP.

In this scenario, each of these four states was implemented one by one using a smartphone[12] that is connected to an AP and transmitting large enough file during the tests. Accordingly, power consumptions were measured by Monsoon-Power-Monitor[13] [42] per state throughout 100-minute test period, and the results were transferred to Table 4. State_1 represents the baseline idle power consumption of the device since the proposed application is not running. After measuring the baseline idle power of the device, the proposed application is launched and the average power consumption when it is running (State_2 to state_4) is measured.

	State_1	**State_2**	**State_3**	**State_4**
Test duration [min.]	100	100	100	100
Avr. power consumption [W]	0.897	0.905	0.909	0.920
Pow. consumption in unit time [x]	1	1.009	1.014	1.026

Table 4. Additional energy cost that is consumed by the application.

The results show that the amount of additional energy consumed by PowerProfiler is in between 0.9% - 2.6%. If the *StepCounter* has already been initiated by another application, additional energy consumed will be even less. Keeping in mind that the application will be used only when needed, not continually and considering the probability of saving high amount of energy (e.g. 42% in the next section), we believe it is an affordable price to pay.

B. Impact of Handover on Throughput and Power Consumption

Figure 9 illustrates the experimental test-bed environment. In this scenario, there are two APs, each of which is located in different lab rooms[14]. These two APs and any device in between were able to reach the signals of each other. Our smartphone, which was programmed to download small amount of packets continuously, first moved to the room where AP_1 is located for 10 minutes. Then, it is moved to the room where AP_2 is located for the next 10 minutes. Same steps were repeated for the next 20 minutes, so that the device visited both AP_1 and AP_2 for 20 minutes, which makes 40 minutes of test period on total.

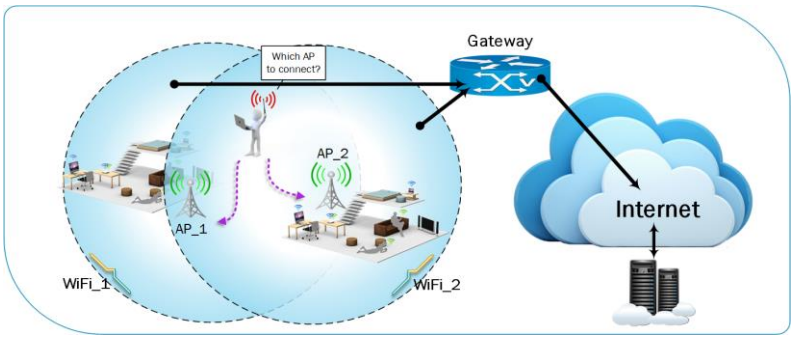


Figure 9. Handover-based network communication scenario.

When the application was off, the device always associated with the AP_1 during the whole test period. The device achieved 1.97 Mbps average throughput when it was in the room where AP_1 is located. Besides, it achieved 0.84 Mbps average throughput when it was in the room where AP_2 is located. Accordingly, average throughput for the device was calculated

[12] A fully-charged LG G3 smartphone was used during the tests. Screen was off and no additional app was in use in the beginning of all tests.

[13] It is a Hardware&Software based solution that enables user to collect power consumption information from the device, bypassing the battery.

[14] AP_1 has one station downloading a large file, and is located in Networks Lab. AP_2 is located in Security Lab across the Networks Lab, and has no station connected.

as 1.47 Mbps. In contrast, when the application was on, the device associated with the AP_1 for the first 10 minutes, and then performed three handovers to the AP_2, AP_1 and AP_2, respectively for the next 10 minutes of time periods. This time, the device achieved 1.95 Mbps average throughput when it was in the room where AP_1 is located, and 3.68 Mbps average throughput (since AP_2 has no additional device connected to itself) when it was in the room where AP_2 is located. Consequently, average throughput for the device was calculated as 2.74 Mbps. Figure 10 illustrates measured results of energy consumption in unit-time, in unit-throughput[15] and average throughput rates, when the application is on and off.

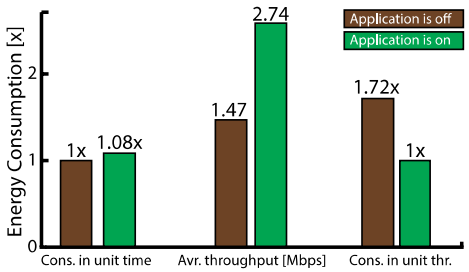
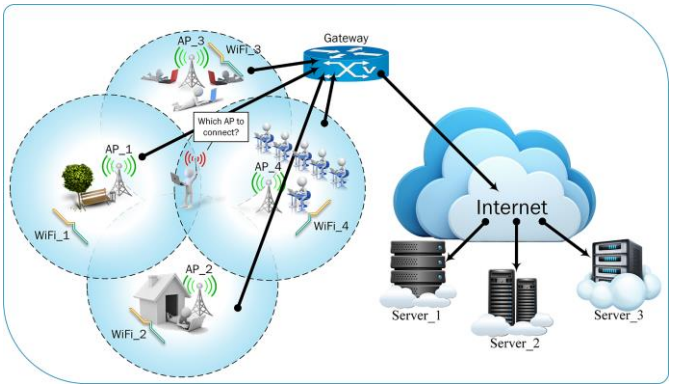


Figure 9. Power consumption metrics when the application is on and off.

The results show that when the application is off, the device requires 1.72 times more energy to receive the same amount of throughput. In other words, the proposed application consumes 42% less energy to receive same amount of data. Hence, it is clear to say that if there are more than one AP in a neighborhood and devices are not stationary, initiating a handover can radically increase both energy efficiency and throughput performance.

C. Impact of AP Channel Utilization on Power Consumption and Throughput

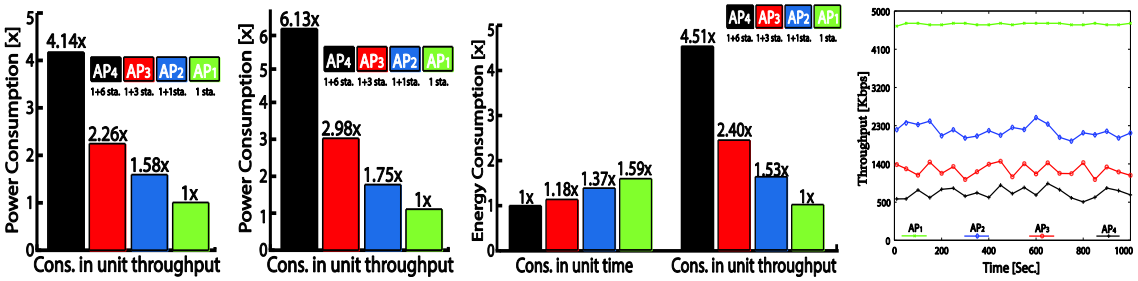
The experimental test-bed environment of this scenario is illustrated in Figure 11, where there are 4 APs, each of which has different amount of traffic (AP_1 has no station connected to itself, AP_2, AP_3 and AP_4 have 1, 3 and 6 stations connected to themselves, respectively and all of these stations are downloading a large file from three different servers installed) and there is one device (LG G3 smartphone) in the center that is implemented with the proposed algorithm and is looking for an AP to associate with. Also note that all APs are equally distant to the device in this scenario. Through this scenario, we aim to examine the impact of AP channel utilization on power consumption and throughput.



[15] Energy consumed to transmit/receive the same amount of throughput. It is computed utilizing the consumption in unit time and the throughput rates.

Figure 11. IEEE 802.11 network communication scenario.

Within this scenario, while the proposed PowerProfiler and the PowerTutor applications are running on the device, the device was provided to connect all four APs one by one and download large enough data during 1000-second test periods. Accordingly, expected amounts of power consumption of the device, which is computed by the PowerProfiler and PowerTutor, when connected to each of the aforementioned four APs separately are shown in Figure 12-a and Figure 12-b, respectively. Measured power consumptions in unit-time and in unit-throughput are shown in Figure 12-c. Finally, actual downlink throughput values of the device after an association with those APs are shown in Figure 12-d.



(a) (b) (c) (d)

Figure 12. Empirical and application-based power consumption and throughput values for downlink flow. (a) PowerProfiler consumption results, (b) PowerTutor consumption results, (c) Measured energy consumptions, (d) Throughput of the device per AP

Results generated by the PowerProfiler (Figure 12-a) shows that connecting to the AP_1 that has no station would be the best choice both for energy and throughput efficiency, as it is expected to save 1.58, 2.26 and 4.14 times more power to transmit the same amount of throughput compared to the association with the AP_2, AP_3, and AP_4, respectively. The results of the PowerTutor are also similar to those of the PowerProfiler, as shown in Figure 12-b. In order to compare the two proposed applications with the actual power consumption values, the power consumed by these applications was also measured using the Monsoon-Power-Monitor and the results are shown in Figure 12-c. In this scenario, PowerProfiler deviates an average of 4.59% compared to actual values, and PowerTutor shows a deviation of 14.61%. Therefore, empirical results validate that PowerProfiler provides the device with more realistic power consumption estimation than PowerTutor.

It should be noted that PowerTutor focuses on how many packets are sent/received at intervals of one second to understand which state the device is in. In PowerTutor, if the number of packets sent/received exceeds a certain threshold, the power consumption state of the device is raised to a higher state. Since the calculations are carried out considering that the device has stayed in that state for one second, it results in the power consumption estimation to be higher than expected. Conversely, PowerProfiler performs a separate calculation for each packet reception/transmission and the results are generated by processing at intervals of 10 milliseconds. Thus, within one-second time slots, how much the device remains in the idle, receiving and transmission states can be computed by PowerProfiler.

The device was also provided to connect all four APs one by one and this time upload large enough data during 1000-second test periods. Expected amounts of power consumption of the device, which are computed by the PowerProfiler and PowerTutor, when connected to each of the aforementioned four APs separately are shown in Figure 13-a and Figure 13-b, respectively. Measured power consumptions in unit time and in unit throughput are shown in Figure 13-c. Actual uplink throughput values of the device after an association with those APs are shown in Figure 13-d.

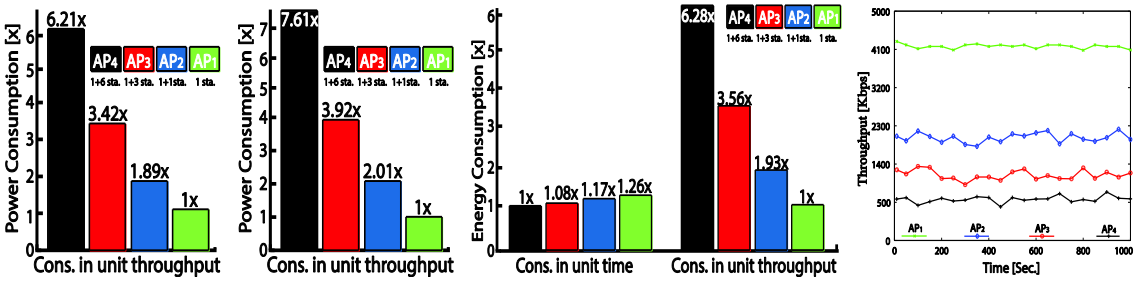


Figure 13. Empirical and application-based power consumption and throughput values for uplink flow. (a) PowerProfiler consumption results, (b) PowerTutor consumption results, (c) Measured energy consumptions, (d) Throughput of the device per AP

Similarly, in this scenario, power consumption estimation generated by the PowerTutor shows a higher deviation (7.65% on average) than the PowerProfiler (1.83% on average) when compared to actual data. The reason for the decline in the amount of deviation is that the device consumes significantly less power in the receiving state than the transmission state.

D. Impact of Signal Strength and RATs on Throughput and Power Consumption

The experimental test-bed environment for this scenario is illustrated in Figure 14, where there are two APs, each of which has one station (downloading a large file) connected to itself. While AP_1 is positioned (in line of sight) very close (less than 5 meters) to the user, AP_2 is positioned (not in line of sight) very far (more than 40 meters) to the user. Apart from the APs, there is one cellular Base Station (BS) available, and there is also one device (LG G3 smartphone) running the proposed algorithm and looking for a PoA to associate with.

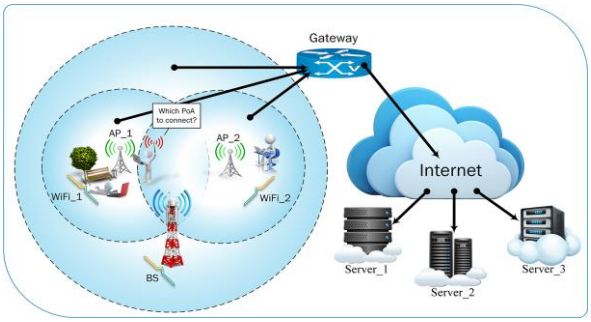


Figure 14. Heterogeneous network communication scenario.

Expected amounts of power consumption of the device computed by the PowerProfiler and PowerTutor, when connected to each of the aforesaid 2 APs and one BS separately are shown in Figure 15-a and Figure 15-b, respectively. Measured power consumptions in unit time and in unit throughput are shown in Figure 15-c. Finally, actual downlink throughput values of the device after an association with those PoAs during 1000-second test periods are shown in Figure 15-d.

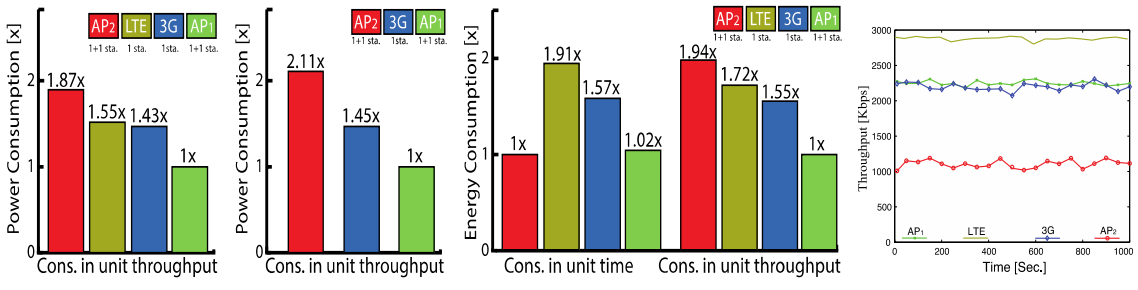
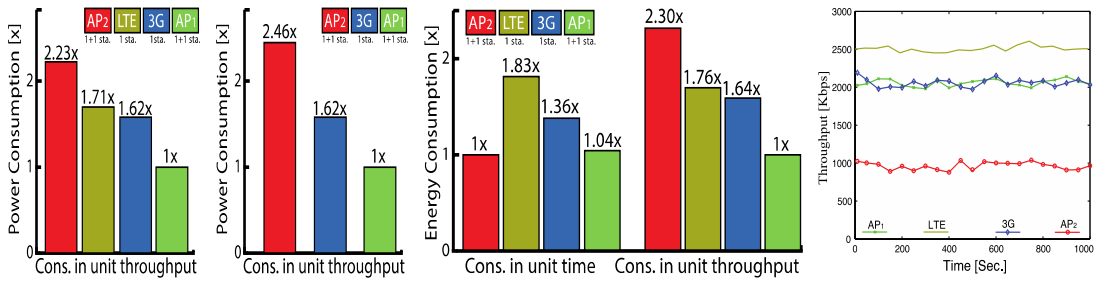


Figure 15. Impact of different signal levels and RATs on throughput and power consumption for downlink flow. (a) PowerProfiler consumption results, (b) PowerTutor consumption results, (c) Measured energy consumptions, (d) Throughput of the device per PoA.

Unless it is highly loaded or having very-low signal strength, packet transmission/reception over the WiFi interface is expected to be more energy-efficient as in cellular networks devices consume power both to maintain their high power states, and to transmit/receive packets in these states. Therefore, AP_1, which is positioned very close (High RSSI) to the device, is the most energy-efficient communication medium compared to the AP_2, 3G and LTE, as shown in both Figure 15-a, Figure 15-b and Figure 15-c. Results generated by the PowerProfiler (Figure 15-a) shows that connecting to the AP_1 is expected to save 1.87, 1.55 and 1.43 times more power to transmit the same amount of throughput compared to the association with the AP_2, LTE and 3G, respectively. Additionally, as seen in Figure 15-d, signal degradation causes a massive drop in throughput and a significant rise in power consumption (since transmitting same amount of data in an error-prone channel takes more time) for the AP_2. Due to the low signal quality, AP_2 is actually expected to consume even more power than the LTE and 3G cellular networks.

In this scenario, PowerProfiler deviates an average of 4.04% compared to actual measured values, and PowerTutor shows a deviation of 4.98%. Since both PowerProfiler and PowerTutor perform state transitions by looking at the inactivity timers and packet transmission/receiving rates, the estimated power consumption values generated by these applications are very close to each other. It should be noted that, unlike PowerTutor, PowerProfiler is able to compute the expected LTE power consumption. As shown in Figure 15-a and Figure 15-c, LTE interface of the device consumes a little more power than the 3G interface, since the tail states (Short DRX and Long DRX) in LTE remain at the higher base power, while much of the 3G tail is in the Cell_FACH state which requires roughly half power [32].

The device was also provided to connect all PoAs one by one and this time upload large enough data during 1000-second test periods. Accordingly, expected amounts of power consumption of the device, which is computed by PowerProfiler and PowerTutor, when connected to each of the aforementioned PoAs separately are shown in Figure 16-a and Figure 16-b, respectively. Measured power consumptions in unit time and in unit throughput are shown in Figure 16-c. Finally, actual uplink throughput values of the device after an association with those APs are shown in Figure 16-d.



(a) (b) (c) (d)

Figure 16. Impact of different signal levels and RATs on throughput and power consumption for uplink flow. (a) PowerProfiler consumption results, (b) PowerTutor consumption results, (c) Measured energy consumptions, (d) Throughput of the device per PoA.

Similarly, in this scenario, power consumption estimation generated by the PowerTutor shows a higher deviation (2.57% on average) than the PowerProfiler (1.45% on average) when compared to actual measured data. Again, the reason for the decline in the amount of deviation is that the device consumes significantly less power in the receiving state than the transmission state in both wireless and cellular interfaces.

VII. CONCLUSION

This paper proposes a real-time network power consumption profiler and an energy-aware network selection tool, which is simply called as PowerProfiler, for Android OS-based smartphones. Android service class and motion detection approach running in background have been updated within this study, so that the application can perform its operations consuming less energy. Moreover, in addition to Wi-Fi and 3G, the LTE interface has also been studied and included into the computations of the proposed solution. In this way, the power monitoring and energy-efficient handover operations are managed more accurately. Finally, user preferences were also added to the application to enable a user-centric increase in energy efficiency. Users can now manage various operations by utilizing the proposed tool such as auto-handover, movement activation, repetition-based scanning, adding already known APs, and setting the threshold values for power-related parameters and step counter.

The proposed tool has been freely released on the Google Play Store. It reports power consumption levels of wireless and cellular network interfaces by utilizing actual packet measurements and precise computations. Extensive experimental test-bed results have been carried out to show the accuracy of the proposed tool compared to others from the literature. The results validate that the proposed tool is very accurate and saves high amount of energy without sacrificing the throughput for any IEEE 802.11 wireless or cellular stations, regardless of having different amount of traffic flow, transmission rates, signal strengths or traffic types.

REFERENCES

[1] Cisco, “Cisco Visual Networking Index: Forecast and Methodology, 2016-2021”, White Paper, June 2017. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf

[2] G. P. Perrucci, F. H. P. Fitzek, and J. Widmer, "Survey on Energy Consumption Entities on the Smartphone Platform," *Vehicular Technology Conference (VTC Spring)*, pp. 1-6, 2011.

[3] R. Kravets and P. Krishnan, "Power management techniques for mobile communication," in *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 157-168, 1998.

[4] G. Anastasi, M. Conti, E. Gregori, and A. Passarella, "Saving energy in wi-fi hotspots through 802.11 psm: an analytical model," *Workshop on Linguistic Theory and Grammar Implementation, ESSLLI*, pp. 24-26, 2004.

[5] *802.11ac: The Fifth Generation of Wi-Fi, August 2012*. Available: http://www.cisco.com/c/ en/us/products/collateral/wireless/aironet-3600-series/white_paper_c11-713103.html

[6] *802.11ac Technology Introduction, March 2012*. Available: https://www.rohde-schwarz. com/us/solutions/wireless-communications/wlan-wifi/in-focus/technology-introduction_106 713.html

[7] A. Enayet, N. Mehajabin, M. A. Razzaque, C. S. Hong, and M. M. Hassan, "PowerNap: a power-aware distributed Wi-Fi access point scheduling algorithm," *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, pp. 1-13, 2016.

[8] Tuysuz M. F., "An energy-efficient QoS-based network selection scheme over heterogeneous WLAN–3G networks." *Computer Networks*, 75, pp. 113-133, 2014.

[9] Tuysuz M. F., and Murat Ucan. "Energy-aware network/interface selection and handover application for android-based mobile devices." *Computer Networks*, 113, pp. 17-28, 2017.

[10] Huaiyu L., Maciocco C., Kesavan V., Low A. L. Y., "Energy efficient network selection and seamless handovers in Mixed Networks", *World of Wireless, Mobile and Multimedia Networks & Workshops, WoWMoM 2009*. pp.1-9, 15-19, 2009.

[11] Bastos J., Albano M., Marques H., Ribeiro J., Rodriguez J., Verikoukis C., "Smart interface switching for energy efficient vertical handovers in ns-2", *Communications, IET* , vol.6, no.14, pp.2228-2238, September 25 2012.

[12] Inwhee J., Kim W. T., Hong S., "A Network Selection Algorithm considering Power Consumption in Hybrid Wireless Networks", *Computer Communications and Networks. ICCCN*, pp. 1240-1243, 2007.

[13] Minji Nam, Nakjung Choi, Yongho Seok, Yanghee Choi, "WISE: energy-efficient interface selection on vertical handoff between 3G networks and WLANs", *Personal, Indoor and Mobile Radio Communications, PIMRC 2004*, pp. 692-698 Vol.1, 5-8 Sept. 2004.

[14] Tran T., Kuhnert M., Wietfeld C., "Energy-Efficient Handoff Decision Algorithms for CSH-MU Mobility Solution", *Computer Communications and Networks (ICCCN)*, July 30 - Aug. 2, 2012.

[15] Petander H., "Energy-aware network selection using traffic estimation", In *Proceedings of the 1st ACM workshop on Mobile internet through cellular networks* (MICNET '09), New York, NY, USA, pp. 55-60, 2009.

[16] Yang W. H., Wang Y. C., Tseng Y. C., Lin B. S. P., "Energy-efficient network selection with mobility pattern awareness in an integrated WiMAX and WiFi network", Int. J. Commun. Syst., 23: 213–230, 2010.

[17] Bastos J., Albano M., Rodriguez J., Verikoukis C., "Location assisted energy efficiency for multi-interfaced mobile terminals", *ICC 2012*, pp. 5660-5664, 10-15 June 2012.

[18] Xenakis D., Passas, N., Di Gregorio, L., Verikoukis, C. "A Context-Aware Vertical Handover Framework Towards Energy-Efficiency," IEEE Vehicular Technology Conference, 2011.

[19] Chowdhury M. Z., Yeong Min Jang, Choong Sub Ji, Sunwoong Choi, Hongseok Jeon, Junghoon Jee, Changmin Park, "Interface selection for power management in UMTS/WLAN overlaying network", *Advanced Communication Technology, 2009. ICACT 2009,* pp. 795-799, 15-18 Feb. 2009.

[20] Calhan A., Ceken C., "Speed sensitive-energy aware adaptive fuzzy logic based vertical handoff decision algorithm", *Systems, Signals and Image Processing (IWSSIP),* 16-18 June 2011.

[21] SungHoon Seo, JooSeok Song, "An energy-efficient interface selection for multi-mode terminals by utilizing out-of-band paging channels", Telecommunication Systems, Volume 42, Issue 1-2, pp. 151-161, October 2009.

[22] Juan Fan, Sihai Zhang, Wuyang Zhou, "Energy-Friendly Network Selection in Heterogeneous Wireless Networks", *Vehicular Technology Conference (VTC Spring),* 6-9 May 2012.

[23] Kolios P., Friderikos V., Papadaki K., "A practical approach to energy efficient communications in mobile wireless networks", Mobile networks and applications, Volume 17, Issue 2, pp. 267-289, April 2012.

[24] SuKyoung Lee, SungHoon Seo, Golmie, N., "An efficient power-saving mechanism for integration of WLAN and cellular networks", *Communications Letters, IEEE* , vol.9, no.12, pp. 1052-1054, Dec. 2005.

[25] Ravi, A., & Peddoju, S. K., Handoff strategy for improving energy efficiency and cloud service availability for mobile devices. Wireless Personal Communications, 81(1), 101-132, 2015.

[26] Choi, J., Lee, G., Shin, Y., Koo, J., Jang, M., & Choi, S. BLEND: BLE Beacon-Aided Fast WiFi Handoff for Smartphones. IEEE International Conference on Sensing, Communication, and Networking, SECON, 2018.

[27] Lee, W., Koo, J., Park, Y., & Choi, S., Transfer time, energy, and quota-aware multi-RAT operation scheme in smartphone. IEEE Transactions on Vehicular Technology, 65(1), 307-317, 2016.

[28] Lee, W., Bae, M., Kim, H., & Kim, H., Mobile device-centric access point monitoring scheme for handover decision triggering in heterogeneous networks. International Journal of Communication Systems, 30(18), e3376, 2017.

[29] L. Zhang, et al. "Accurate online power estimation and automatic battery behavior based power model generation for smartphones." *Hardware/Software Codesign and System Synthesis, CODES+ ISSS*, 2010.

[30] J. Wonwoo, et al. "DevScope: a nonintrusive and online power analysis tool for smartphone hardware components." *IEEE/ACM/IFIP Hardware/software codesign and system synthesis*. ACM, 2012.

[31] Y. Chanmin, et al. "AppScope: Application Energy Metering Framework for Android Smartphone Using Kernel Activity Monitoring." *USENIX Annual Technical Conference*, 2012.

[32] D. Fangwei, et al. "Monitoring energy consumption of smartphones." *International Conference on Cyber, Physical and Social Computing*. IEEE, 2011.

[33] Google Play Store. Intel performance viewer, play.google.com. Accessed December 2016.

[34] Qualcomm Inc. Trepn profiler, From developer.qualcomm.com, Accessed December 2016.

[35] Google Play Store. Gsam battery monitor, play.google.com. Accessed December 2016.

[36] Harjula E., Kassinen O., Ylianttila M., "Energy consumption model for mobile devices in 3G and WLAN networks", *Consumer Communications and Networking Conference (CCNC)*, pp. 532-537, 14-17 Jan. 2012.

[37] AT&T Developer Program, Comparing LTE and 3G Energy Consumption, Available: https://developer.att.com/application-resource-optimizer/docs/best-practices/comparing-lte-and-3g-energy-consumption.

[38] Jensen, Anders R., et al. "LTE UE power consumption model: For system level energy and performance optimization." *Vehicular Technology Conference (VTC Fall)*, 2012.

[39] IEEE Std 802.11, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications", 1999 Edition.

[40] G. Kalic, I. Bojic, M. Kusek, Energy consumption in android phones when using wireless communication technologies, in: MIPRO, 2012 Proceedings of the 35th International Convention, pp. 754–759, 21–25 May 2012.

[41] L. Wang, A. Ukhanova, E. Belyaev, Power consumption analysis of constant bit rate data transmission over 3G mobile wireless networks, in: ITS Telecommunications (ITST), pp. 217–223, 23–25 August 2011.

[42] Lauridsen, Mads, Preben Mogensen, and Laurent Noël. "Empirical LTE smartphone power model with DRX operation for system level simulations." *Vehicular Technology Conference (VTC Fall)*, 2013.

[43] Xia, F., Hsu, C. H., Liu, X., Liu, H., Ding, F., & Zhang, W., The power of smartphones. Multimedia Systems, 21(1), 87-101, 2015.