

On Power and Throughput Tradeoffs of WiFi and Bluetooth in Smartphones

Roy Friedman
Computer Science, Technion
Email: roy@cs.technion.ac.il

Alex Kogan
Computer Science, Technion
Email: sakogan@cs.technion.ac.il

Yevgeny Krivolapov
Physics, Technion
Email: evgkr@tx.technion.ac.il

Abstract—This paper describes a combined power and throughput performance study of WiFi and Bluetooth usage in smartphones. The study reveals several interesting phenomena and tradeoffs. The conclusions from this study suggest preferred usage patterns, as well as operative suggestions for researchers and smartphone developers.

I. INTRODUCTION

Smartphones are quickly becoming the main computing and (data) communication platform. These days, smartphones are all equipped with Bluetooth and WiFi, which complement their cellular communication capabilities. Bluetooth was originally placed in mobile phones for personal-area communication, such as wireless earphones, synchronization with a nearby PC, and tethering. WiFi was added more recently in order to improve the users' Web surfing experience whenever a WiFi access point is available. In fact, new market researches predict that between 2012 and 2014, depending on the source, WiFi equipped smartphones will outnumber all other WiFi enabled devices combined (laptops, tablets, WiFi enabled TVs, etc.).

When examining the anticipated usage patterns of WiFi on smartphones, it appears that in addition to fast and possibly free Internet access (including VoIP and video), direct communication between nearby devices is of growing interest. Obvious examples include media streaming either between smartphones or between a phone and another nearby wireless device (TV or computer) in a home or office setting. This is exemplified in the plethora of new WiFi based streaming solutions, as well as being one of the main motivations behind the WiGig initiative. Another scenario includes ad-hoc social networking and communication, such as iPhone's iGroups, Nokia's Instant Community, Mobiluck, and WiPeer, to name a few.

Such local communication can be potentially performed either over WiFi or over Bluetooth. Some could be performed while relying on a mutual nearby access point (AP), while others might be more natural for WiFi ad-hoc mode. When considering these alternatives, important considerations include the obtainable throughput and power consumption of each. As Bluetooth was planned for personal area communication, its

transmission range is much shorter than WiFi. Hence, comparing the two is meaningful only when the devices are close enough so that both can be used. Given that Bluetooth also offers lower bandwidth, one might ask why bother with it at all. The reason to still consider Bluetooth is that previous power studies have suggested that Bluetooth is much more power efficient than WiFi [1]–[4], which motivated multiple works on clever combinations of both Bluetooth and WiFi [1], [3], [5]–[7]. This issue is acute given that wireless communication is one of the main battery drainers in a mobile phone [1], [8].

It is important to notice that both a Bluetooth interface and a WiFi interface can be in multiple modes, or states, such as transmitting, receiving, connected but idle, disconnected and hidden, disconnected and seeking, etc. (the exact names in each technology are specified later in this work); each of these states may consume different power levels. One of the main criticism of WiFi in past works has been that its power consumption in idle mode is on the same order of magnitude as sending and receiving [1], [8]–[11]. Also, as we just mentioned, Bluetooth was reported to be at least an order of magnitude more power efficient than WiFi [1]–[4]. Yet, when examining these works, we have discovered some shortcomings, which have motivated this study: First, many of these studies were performed several years ago, and given the rapid change in technology, one cannot trust that they remain valid. Second, in most of these studies, each interface was measured independently and in isolation in a lab setting. Hence, it is not clear that the results hold for a mobile phone, in which the two RF interfaces are packed in close proximity, sometimes on the same chip. Moreover, from the user's point of view, it is interesting to know the total power consumed for a given operation due to such communication, including, e.g., code that might be running in the Kernel of the OS, outside the wireless interface. Finally, some of the more recent works that did measure modern smartphones, either did not examine the different states of WiFi and Bluetooth, focused only on WiFi, and/or did not look at the interplay between bandwidth and power.

In this work, we present a power and throughput study of WiFi and Bluetooth on modern smartphones, in which we investigate the relationship between obtained throughput and power, and the power consumption in the various states of

This work was supported by the Israeli Science Foundation grant No. 1247/09 and by Microsoft R&D Labs, Israel. The work of A. Kogan was also supported by the Technion Hasso Plattner Center.

the wireless interfaces. In the process, we discover several interesting phenomena (some counter previous conventions) and draw some practical recommendations for future academic research and practical development. Some of the highlights of our findings include:

- 1) For WiFi, the power consumption is generally linear with the obtained throughput. However, whenever the sender's network card (NIC) is more capable than that of the receiver, there is a threshold point at which the power consumption doubles, while the throughput continues to grow linearly. Beyond this point, when running over UDP and trying to further increase the offered load, the obtained throughput eventually starts to drop as well, while power consumption continues to rise. When running over TCP, which inherently self regulates its offered load, the obtained throughput is slightly beyond the threshold point. This means that TCP stops too late; if it would have constrained the sender's rate to be slightly lower, it could have saved half the energy! We discuss this phenomenon, explain why it happens, and suggest ways of eliminating it in TCP (Section IV-B).
- 2) For WiFi, we discovered that when the phones are connected to an access point but not communicating, their WiFi related power consumption is marginal. This means that the Power-Saving Mode (PSM) of WiFi is highly effective. Consequently, unless a phone is communicating, as long as it is connected to an access point, WiFi will hardly drain its battery (Section IV-A).
- 3) For WiFi, when phones are in ad-hoc mode, the power cost is on the same order of magnitude as send and receive. Also, the power cost is high when WiFi is disconnected and searches for available networks (in one of the tested smartphones, it is as high as in the idle ad-hoc mode). This means that in order for WiFi transmitters to be constantly left on, and in order to convince people to use ad-hoc networks, the power consumption of these two modes needs to be improved. In particular, ad-hoc PSM should be implemented for ad-hoc mode and more efficient AP seeking algorithms should be employed (Section IV-A).
- 4) For Bluetooth, the power consumption while sending and receiving is lower than WiFi, but only by about a half. This counters previously published results [1], [2], [4]. We discuss possible reasons for this and derive a formula that combines the power and obtained throughput in order to determine which technology should be used in order to transfer a file of a given size. Based on the performance numbers we have obtained, it appears that for virtually all interesting file sizes, it is always better to use WiFi (Sections IV-B and IV-C).
- 5) When combining Item 2 with Item 4, we can conclude that once WiFi is connected to an AP, there is no benefit in using Bluetooth, as opposed to the work in [1]. More-

over, if PSM would be implemented for ad-hoc mode, and smarter AP seeking algorithms be employed, then WiFi will always be preferable over Bluetooth (Section V).

- 6) We also show some differences in performance based on the memory type to which received files are being stored, and discuss the operating systems' related reasons for it (Section IV-C).

Our work was performed using the following platforms: Samsung Omnia i900 running Windows Mobile 6.1, HTC Diamond 2 running Windows Mobile 6.5, Samsung Galaxy running Android 1.5, and Samsung Spica running Android 2.1. The differences in the performance results, as measured across these platforms, expose different design choices taken mostly by the OS and driver developers, as we discuss in relevant parts of the paper. At any event, none of the phones can be claimed to be better than the others in the parameters we have measured; indeed, our intention was not to find a winner, but rather to offer insight on such issues and hopefully guidelines for future phone developers.

II. RELATED WORK

Energy efficiency is recognized as a paramount property of any mobile device, in particular for smartphones. Consequently, several papers try to address the question of where and how the energy is consumed. Carroll and Haiser [12], for example, design a set of micro-benchmarks to independently associate the power costs with a particular part of the smartphone system, such as CPU, RAM, WiFi, Audio, etc. in a Openmoko Neo Freerunner phone. While their work considers multiple wireless interfaces available on the devices under test, the interfaces are not profiled beyond enabled and disabled mode, and only their power cost in various general benchmarks is considered.

Few recent works examine energy consumption of the WiFi interface in mobile phones. Balasubramanian et al. [13] compare between cellular interfaces, such as 3G, GSM, and WiFi. They utilize OS-specific methods to collect information on power usage, such as Nokia Energy Profiler. In comparison, our framework is generic and can be used virtually with any mobile phone. Additionally, the (802.11b) WiFi interface considered in [13] is outdated and is profiled only in access point mode of operation. In another work, Xiao et al. [11] model and measure the power consumption of 802.11g WiFi interface on three mobile phones. Their experiments also consider only access point operation mode and are conducted with TCP traffic only.

Power measurements of a WiFi card in various modes are reported in [9], [14] (more than a decade ago). Unlike these works, we have chosen to measure the performance of wireless interfaces that are integrated on-board. We believe that for an end-user, the performance of the integrated radio in different network settings (including various hidden costs, e.g., running kernel code and copying buffers) is even more relevant than the performance of a particular wireless card.

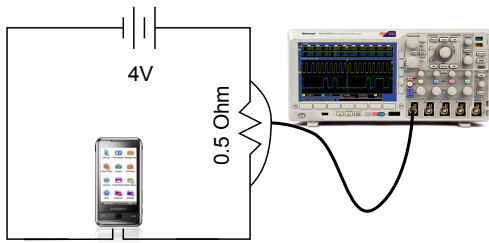


Fig. 1: The circuit used to determine the power consumption.

Although several papers explore trade-offs between WiFi and Bluetooth interfaces in mobile devices (e.g., [1], [3], [6], [7]), with the exception of [1], none of them actually measures the performance of the latter. In fact, we are not aware of any such study for smartphones. Several other works consider the power cost of Bluetooth in the context of sensor networks [2], [4]. Interestingly, the ratio of power cost for transmission to the idle state, as specified by these papers, is much lower than actually measured in our experiments. As elaborated in Section V, this result may have an important implication on the design of multi-radio protocols and applications, such as energy-aware file transfer.

III. EXPERIMENTAL SETUP

We have employed a circuit as presented in Fig. 1. The circuit was fed by a constant voltage, V_0 , using an HP E3620A DC power supply, and the current in the circuit was determined by measuring the voltage across a resistor that was connected serially to the phones. The phones were connected through pin contacts normally used to connect a battery. We have used a Tektronix DPO3034 digital oscilloscope to acquire and integrate the signal across the resistor. The voltage across the connected phone is given by

$$V_{ph}(t) = V_0 - V_R(t), \quad (1)$$

where $V_R(t)$ is the voltage across the resistor and $V_0 = 4 \pm 0.01V$ is the feeding voltage (for the Diamond phone, we used $V_0 = 4.3 \pm 0.01V$, since $4V$ was insufficient for its operation. We also tested other phones with $V_0 = 4.3V$ and did not find any significant difference in performance results). We have selected a resistor of low resistance, $R = 0.5 \pm 0.05\Omega$, in order to assure small fluctuations of $V_{ph}(t)$. Since the maximum achieved current in the circuit in all the experiments is $I_{max} = 500$ mA, the maximal voltage fluctuation is 250 mV. The current in the circuit is given by

$$I(t) = V_R(t)/R, \quad (2)$$

and therefore the instant power consumption of the phone is

$$P_{ph}(t) = I(t) \cdot V_{ph}(t) = \frac{V_R(t)}{R} (V_0 - V_R(t)). \quad (3)$$

It is generally hard to measure specific power events, such as sending a single packet over a wireless link, since one has

to synchronize the oscilloscope to the event. To avoid this difficulty, we have not measured the power consumption due to specific power events, but rather integrated it over a large interval of time where this event has occurred many times. We then calculate the average power consumption,

$$\bar{P}_{ph} = \frac{1}{T} \int_0^T P_{ph}(t') dt', \quad (4)$$

and also the standard deviation from the average,

$$d\bar{P}_{ph} = \frac{1}{T} \int_0^T (P_{ph}(t') - \bar{P}_{ph})^2 dt', \quad (5)$$

where T is the integration time. The integration was long enough, such that the average power was stationary and its standard deviation was low enough. Typically, T was of the order of few seconds. Both the average power and its standard deviation were calculated directly by the oscilloscope.

When setting up the experiments, we have observed that the power consumption of the base setting, in which the phone has all radios turned off, may vary significantly. These variations are attributed to the phone's touch-screen brightness, which stays intact through the same set of experiments, but may change when the phone is rebooted. To eliminate this effect, we have measured the base power consumption at the beginning of each experiment and subtracted this value from the power numbers measured in each setting. Thus, the reported numbers have the form $x \pm y$, where x is the additional power consumed by phones in various settings as an offset from a measured base and y is the standard deviation of that measurement.

Throughput was measured in several ways. First, we have employed `iperf`, a standard network performance measurement tool [15], which we have adapted to run on mobile phones and to support Bluetooth [16]. Second, we have utilized standard file transfer applications and measured the time to transfer large files; for Bluetooth, it is the built-in file transfer feature available on Windows Mobile and Android, whereas for WiFi, FTP software was employed. It should be noted that while power is averaged over time, throughput is averaged over a number (10) of experiments. Although in some configurations the throughput measurements exhibited some variance (as reported in the paper), the instant power was consistent during all throughput measurements, exhibiting only negligible differences. Thus, the reported power is of the first throughput experiment performed for each particular configuration.

IV. EXPERIMENTAL RESULTS

A. Power in Non-Communicating Modes

Table I presents the power consumption of the Bluetooth and WiFi radios in modes of operation that do not involve communication performed by a user. We refer to these modes as *non-communicating*, although in some cases, radios may transmit some packets (e.g., when searching for networks).

Configuration		Galaxy	Spica	Omnia	Diamond
base (all radios off)		0 ± 48	0 ± 8	0 ± 2	0 ± 12
BT	discoverable	88 ± 56	136 ± 16	29 ± 3	60 ± 144
	non-discoverable	8 ± 32	0 ± 16	29 ± 3	60 ± 144
	scanning	160 ± 48	152 ± 16	173 ± 8	676 ± 2
WiFi	not connected	72 ± 64	32 ± 48	2 ± 2	60 ± 72
	searching	72 ± 64	32 ± 48	661 ± 8	60 ± 72
	connected to AP	32 ± 24	32 ± 40	21 ± 2	44 ± 80
	connected to AH	N/A	N/A	664 ± 2	1284 ± 6

TABLE I: Power consumed by Bluetooth and WiFi radios in different scenarios, as an offset from a measured base.

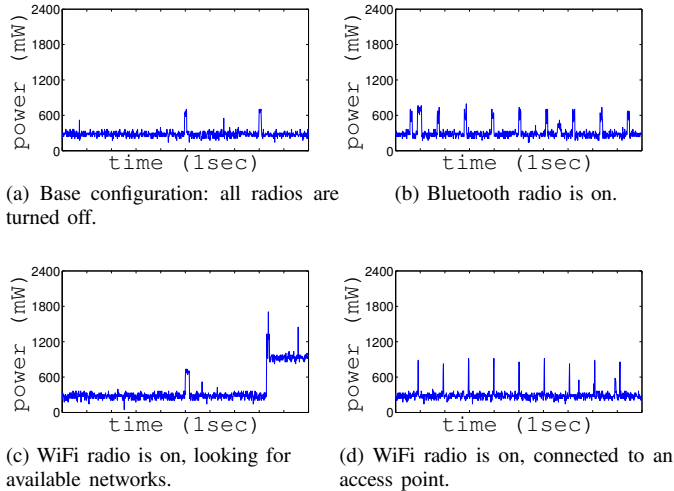


Fig. 2: Instant power consumed by the Omnia phone.

In the base configuration, when all radios are off, all phones except for Galaxy have a fairly stable power consumption. Power fluctuations in the base configuration, reflected by the standard deviation, can be probably associated with the periodic scanning of the touch-screen performed by the phones. These fluctuations in the form of small peaks can be seen on Fig. 2a produced for the Omnia phone. Our hypothesis is supported by the fact that the peaks are gone when the screen is turned off.

Bluetooth: We consider three configurations of the non-communicating Bluetooth radio: (1) discoverable mode, where other Bluetooth-equipped devices may discover the phone when inquiring for nearby located devices, (2) non-discoverable mode, and (3) scanning mode, when the phone searches for other nearby devices. In the discoverable mode, the additional power is required to listen to the inquiry requests sent by nearby devices. Still, the consumption of the Android-based phones is considerably higher than that of Windows Mobile-based. On the other hand, while the consumption of the Windows Mobile-based phones remains the same in both discoverable and non-discoverable modes, the Android-based phones almost do not use any additional power in the non-discoverable mode. Fig. 2b shows the instant power consumption of Omnia when its Bluetooth is turned on. It is worth noting that the Android-

based phones limit the duration of the discoverable mode to 120 seconds, after which the phones automatically switch to the non-discoverable mode, while the Windows Mobile-based phones may stay in the discoverable mode with no limit.

These findings suggest that the two operating systems employ different approaches to Bluetooth radio management. The Windows Mobile-based phones operate the radio in the same state, requiring same additional power, without relating to whether the device was set as discoverable or not. Differently, the Android-based phones are optimized to change the physical state of the radio based on the configuration of the ability to be discovered. In the common case, Bluetooth discovery is required rarely and for certain periods of time (e.g., when a user wishes to connect to the phone from her laptop or earphone, etc.). Thus, in the long run, the approach employed by the Android-based phones is more power-efficient than the one employed by the Windows Mobile-based phones.

We should note the high fluctuations of power costs in both discoverable and non-discoverable modes as measured with the Diamond phone. Although the Galaxy phone also has an unstable power cost, the standard deviation reported for the Diamond phone is exceptionally high. We did not find any acceptable explanation for this phenomenon, except for noticing that in several other configurations as discussed below, Diamond also has much worse power management when compared to other tested smartphones.

In the scanning mode, all phones, except for Diamond, consume a similar amount of additional power, while this amount (for all phones) is considerably higher than in other previously discussed modes. The power consumption of Diamond is outstandingly high. This could be related to the particular chip design on which the Bluetooth radio is located. Based on these results, we conclude that the scanning mode of Bluetooth in smartphones is very power-demanding and should be used very selectively. Several other works [4], [7] confirm our findings about these differences.

WiFi: For WiFi radio, we consider four non-communicating modes: (1) not-connected mode, where the radio is on, but not connected to any network, (2) searching mode, in which the radio scans for available networks, (3) connected to access point mode, and (4) connected to ad-hoc network mode.

The clear distinction between the first two modes exists only at the Omnia phone. In this phone, the WiFi radio in the first mode consumes almost no additional power, while in the second mode the consumption increases drastically. We were able to determine these modes by turning the access point on and off and noticing that the phone updates the availability of the access point only when its power consumption was raised. The transition between these two modes is caught in the right side of Fig. 2c. We note that we did not find neither regularity in the time intervals spent by the radio in each of the states nor any way of configuring these intervals.

On the other hand, the other phones do not show any

distinction between these two modes. Their power consumption remains roughly the same over time, while the change in availability of the access point is reflected almost immediately by the phones without causing any rise in power consumption. These results expose the tradeoff between power consumption and the freshness of the list of available access points. Determining which phone has a better power strategy when WiFi is unconnected is somewhat ambiguous, and depends on the exact pattern of the searching/idle intervals employed by Omnia.

When connected to the AP, all phones automatically switch into power-saving mode (PSM). This mode appears to be very power-efficient, as it significantly reduces the power consumed by the phones, making it comparable with the power consumed by non-communicating Bluetooth radio. In PSM, WiFi radio sleeps most of the time, waking up periodically to receive a beacon from an AP, which indicates the phone whether the AP has a pending traffic for it (for more details on the 802.11 PSM, refer to [17]). Fig. 2d shows the plot of the instant power consumed by the Omnia phone when connected to the AP. The highest spikes relate to the power required to receive beacons sent by the AP every 100ms.

Since Android-based phones lack official support for the ad-hoc mode, the power cost of WiFi radio in this mode was measured only for Windows Mobile-based phones. The results show that the power consumption of an idle WiFi radio connected to an ad-hoc network is much higher than in the access point case. This is because no default PSM is employed when the radio operates in ad-hoc mode. The significant power consumption of an idle radio connected to an ad-hoc network was also reported in previous works (e.g, [9]). Once again, the power cost of the Diamond phone in this mode is extremely high compared to that of Omnia.

B. Measurements with *iperf*

In this section, we report on results achieved with *iperf*, a standard tool for network performance measurements [15], which we have ported to Windows Mobile and extended to support Bluetooth communication. The technical details of these enhancements can be found in [16]. As *iperf* is not yet ported to Android, the experiments with this tool were conducted only on the Omnia and Diamond phones.

1) *Max Throughput Experiments*: For WiFi, we experimented with both UDP and TCP. For Bluetooth, we experimented with RFCOMM (the only protocol supported by *iperf* on Windows Mobile for Bluetooth communication [16]. This is not a real limitation, though, since RFCOMM, being a robust general-purpose and reliable protocol, is the primary choice for any application requiring Bluetooth communication). In all experiments, we have used a Lenovo T61 laptop running Windows XP Service Pack 3 as a corresponding peer for communication with the phones. In addition, for access point network configurations, we used a Linksys WRT54GL wireless router. In all experiments with UDP, *iperf* was configured

to send packets at a rate of 54Mb/s, which is the theoretical 802.11g link bandwidth. This is in order to achieve the maximal available link throughput. In the case of TCP and RFCOMM, *iperf* always strives to achieve the highest possible rate, which is ultimately controlled by these protocols' internal congestion and flow control mechanisms.

Table II shows the results: the throughput in KB/s, the corresponding power in mW, the *utility* value – the throughput per power in KB/mW·s, and the percentage of message loss (as reported by *iperf* for UDP). Clearly, the utility value indicates which protocol is more efficient in terms of the amount of energy it consumes per bytes transmitted (or received).

The results in Table II expose four interesting phenomena: First, Bluetooth communication in smartphones consumes a significant amount of energy (compared to non-communicating states reported in Table I), much higher than previously reported. For example, Pering et al. [1] reported a factor of 4.8 whereas we found a factor of 15.7 for receiving and 17.9 for sending in the case of Omnia. In particular, the power consumption of Bluetooth is only 2–3 times lower than WiFi. This contradicts previously published results [1], [2], [4] that reported differences in orders of magnitude. We discuss the implications of this in Section V.

Second, when examining the utility of each protocol, in the Omnia phone, the utility value of Bluetooth RFCOMM is roughly 3 times lower the one obtained with WiFi in either protocol (except for receiving in UDP, as discussed below). For the Diamond phone, the utility value for Bluetooth is even worse; 4 to 5 times lower than WiFi. Consequently, WiFi is a much more power-efficient way of communication than Bluetooth. Hence, Bluetooth usage should be restricted to situations where the corresponding peer does not support WiFi, as in the case of earphones. This outcome also seriously questions the need for solutions like those proposed in [1], where Bluetooth is used to carry traffic between Bluetooth-enabled access points and mobile devices.

Third, the utility value for ad-hoc networks is similar to that of access point networks. This happens even though in ad-hoc mode, there is a direct communication between a sender and a receiver, while in access point mode, the communication goes through a third party. The difference in the performance is not observed since the actual link throughput achieved by smartphones and by the laptop is much lower than the maximal theoretical bandwidth of the link (54Mb/s).

Finally, notice that the receiver's throughput in the case of UDP using Omnia is much lower than the sender's throughput. This is due to the fact that the WiFi card of the laptop can send data at a higher sustainable speed than the card of the mobile phone is capable of receiving. Thus, many transmitted packets are dropped by the phone, as confirmed by the high numbers of lost messages reported in Table II for this case.

2) *Optimizing Throughput-to-Power Ratio*: Next, we would like to explore more thoroughly the tradeoff between through-

		Omnia					Diamond				
		Bluetooth	WiFi				Bluetooth	WiFi			
			Access point		Ad-hoc			Access point		Ad-hoc	
		RFCOMM	TCP	UDP	TCP	UDP	RFCOMM	TCP	UDP	TCP	UDP
sending from phone	KB/s	137	1186	1136	1232	1075	115	1041	969	1034	978
	mW	520	1568	1544	1600	1560	748	1530	1524	1548	1548
	KB/mWs	0.26	0.76	0.74	0.77	0.69	0.15	0.68	0.64	0.67	0.63
	% lost			0.4		0			0.4		0
receiving by phone	KB/s	128	1201	627	1336	618	135	1345	1446	1294	1223
	mW	456	1504	1496	1496	1448	708	1484	1468	1512	1460
	KB/mWs	0.28	0.80	0.42	0.89	0.43	0.19	0.91	0.99	0.86	0.84
	% lost			58		55.9			4		2.3

TABLE II: Power consumed (as an offset from a measured base) and throughput obtained by Bluetooth and WiFi radios of the Windows Mobile-based phones when sending and receiving data with `iperf` under various transport protocols.

put and power. To simplify the presentation, we refer to the numbers of ad-hoc mode while mentioning the numbers for access point mode in parenthesis. We start by varying the sender's bandwidth limitation of `iperf` when running with UDP and examining the obtained throughput and power consumption at Omnia. The results are shown in Fig. 3a and 3b.

As can be seen, until the sender's rate limitation reaches approximately 10 Mb/s (9.75 Mb/s), the throughput offered by the sender is almost equal to the throughput obtained by the receiver, indicating near zero message loss. At the same time, the power consumed by the receiver grows linearly with the throughput, meaning a roughly constant throughput-to-power ratio. However, between 10 and 10.25 Mb/s (9.75 and 10 Mb/s), a significant spike in power consumption occurs without any significant message loss (at the IP level). Yet, beyond 10.25 Mb/s (10 Mb/s), the receiver's bandwidth drops dramatically (message loss grows dramatically) while the power consumption continues to climb linearly.

We believe the reason for this strange behavior is as follows: When the sender's limit reaches 10.25 Mb/s (10 Mb/s), the MAC (and possibly PHY) layers of the receiver's WiFi card start dropping messages. Yet, given 802.11's retransmission mechanism, such messages are retransmitted at the MAC level. Hence, at the IP level there is no noticeable message loss. However, the WiFi card now has to work much harder for each message it delivers to the IP layer, which explains the spike in power. Beyond 10.25 Mb/s (10 Mb/s), MAC retransmissions are no longer sufficient, and messages start dropping even at the IP level.

The conclusion is that we can significantly improve the throughput-to-power (utility) of UDP by limiting the sender's rate to 10 Mb/s (9.75 Mb/s) or below. As mentioned above, this way, power would not be wasted on messages that cannot be processed by upper layers in the networking stack. Moreover, notice from Tables I and II that the throughput achieved by TCP is around 10 Mb/s, while the power consumption offset from the idle state is 832 mW (1483 mW). These numbers correspond to UDP in the sub-optimal case, i.e., after the spike

in power consumption. This implies that the performance of TCP, as expressed by the utility value, might also be improved if we limit the rate at which the sender transmits data.

Unlike UDP, TCP employs a congestion and flow control mechanism, which provides tools for both the sender and the receiver to control the rate at which data is sent. One of these tools is limiting the size of the receiving window, i.e., the size of the buffer allocated by the operating system of the receiver for the TCP incoming messages. The advantage of this approach is that it does not require any modification of the TCP protocol at the sender, while the receiver is responsible to match the size of its window to its current load and the ability to process incoming messages. The disadvantage is a relatively coarse granularity of the flow control provided by this method, since it controls the maximum number of TCP packets sent per time unit rather than the number of bytes.

In Windows Mobile, the size of the receiving window can be controlled via the registry [18] (and requires an OS reboot after each registry update). The default size for the window in Omnia is 0x8000 (32768) bytes. We experimented with different sizes to estimate the impact of the flow control on the utility value. The results are presented in Fig. 3c and 3d. Notice the rapid increase in power consumption when the window size is increased from 0x2000 to 0x3000 bytes in the ad-hoc case (from 0x6000 to 0x7000 in the access point case). At the same time, the throughput does not increase significantly, staying around 9 Mb/s. After this spike, power consumption stays almost constant, while throughput slowly converges towards the maximal value around 10 Mb/s. This value is achieved with the default TCP receiving window size, while further enlargement of the window has a very small effect on throughput. The higher receiver throughput obtained by TCP compared with UDP is presumably due to TCP's retransmissions.

We would like to mention that we have also gathered supporting evidence from TCP to the fact that the MAC layer starts retransmitting messages at a slightly lower load than when the IP layer starts missing messages. These include an increase of the round-trip delay of TCP as well as the sender's

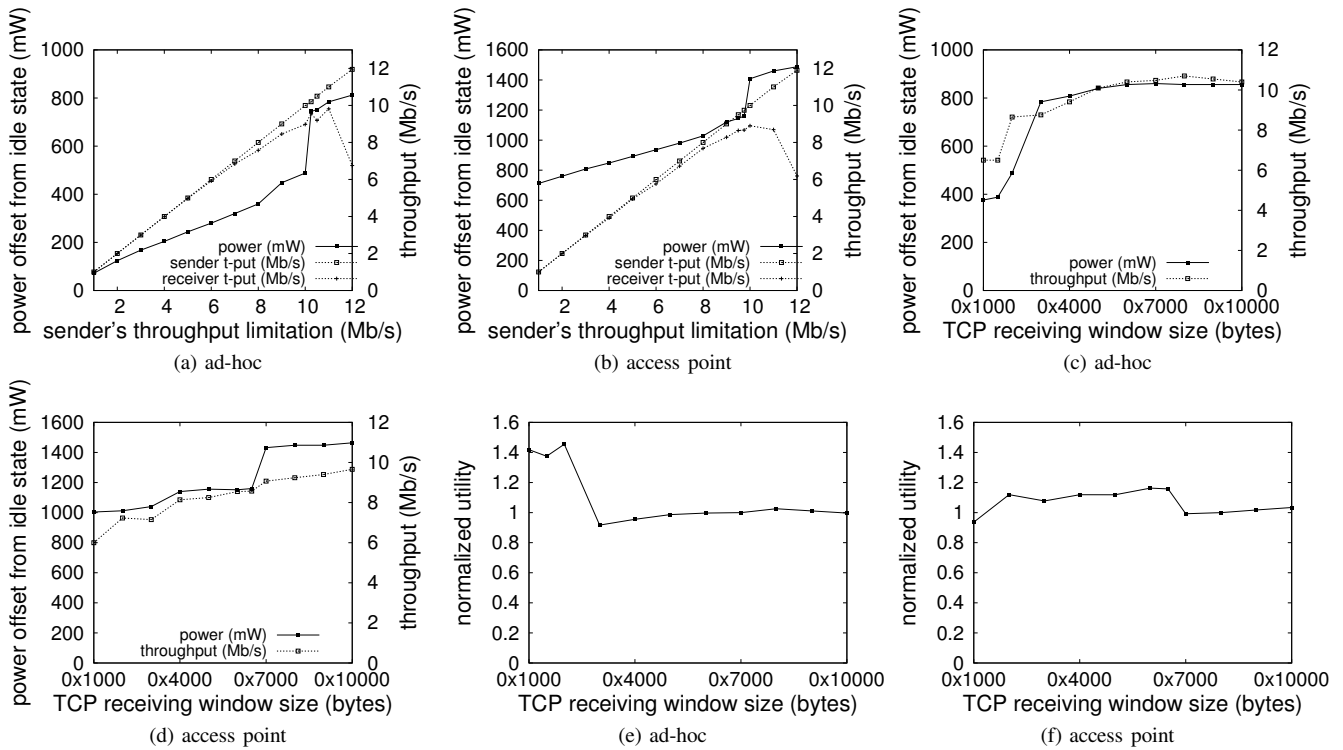


Fig. 3: (a,b,c,d): Power consumed and throughput obtained by the WiFi radio at Omnia when receiving data sent by *iperf* with the UDP protocol (a,b) and TCP protocol (c,d). (e,f): The normalized utility achieved by the WiFi radio at Omnia when receiving data sent by *iperf* with the TCP protocol.

network card reporting an increased number of retransmissions.

Fig. 3e and 3f present the utility value achieved by TCP as a function of the TCP receiving window size. The value is normalized by the result achieved for the default window size (i.e., the value for the window of 32768 bytes is 1). It is essentially the same data presented in Fig. 3c and 3d, but from a different perspective. It clearly shows that the TCP utility value of the Omnia phone operating in ad-hoc mode can be improved by more than 45% by controlling the TCP flow using the approach presented in this paper. For the access point case, the improvement is more than 16%.

Notice that while the optimal window size in the ad-hoc case is 0x2000 bytes, in the access point case it is 0x6000 bytes. This means that the discussed optimization of the throughput-to-power ratio is more than a simple performance tweak, and should be applied dynamically based on the current networking conditions. This observation motivates the development of a dynamic optimization of the throughput-to-power ratio, and is discussed in more details in Section V.

C. Measurements with File Transfer

The results presented in Section IV-B were obtained using *iperf*, which does not access persistent memory for reading or writing transmitted data. Hence, the actual throughput

Configuration		Galaxy	Spica	Omnia	Diamond
sending from root	mW	N/A	584 ± 80	511 ± 18	696 ± 3
	KB/s	N/A	115.7	67.3	63.6
receiving to root	mW	N/A	376 ± 80	489 ± 18	724 ± 24
	KB/s	N/A	112.8	84.9	104.4

TABLE III: Power consumed (as an offset from a measured base) and throughput achieved by Bluetooth-based file transfer.

achieved by a user while sending or receiving a file can be different (and, probably, much lower) due to various delays incurred by internal I/O. In order to complete the picture and validate this assumption, we performed several experiments with file transfers between the smartphones and the laptop.

Due to limitation of the operating system, the Galaxy phone does not support Bluetooth-based file transfer. Thus, the performance of Bluetooth was measured for the other three phones, using file transfer features built in the corresponding operating systems. The performance results of WiFi were obtained using the file transfer protocol (FTP), which utilizes TCP as an underlying transport protocol. For this purpose, the phones were installed with an FTP server (SviFTP for Android-based and Mocha FTP for Windows Mobile-based phones) and the laptop was installed with an FTP client (Core FTP).

We experimented with different types of memory sources

Configuration		Galaxy access point	Spica access point	Omnia		Diamond	
				access point	ad-hoc	access point	ad-hoc
sending from flash	mW	968 ± 32	N/A	1192 ± 8	1144 ± 8	N/A	
	KB/s	861.2		751.2	751.2		
sending from root	mW	992 ± 88	992 ± 64	1572 ± 8	1608 ± 8	1532 ± 3.2	1556 ± 4
	KB/s	830.4	898.4 ± 100.9	887.8	887.8	755.9 ± 55.9	811 ± 16.1
receiving to flash	mW	744 ± 80	N/A	1120 ± 200	1376 ± 200	N/A	
	KB/s	1099.7		243.2 ± 26.2	299.8 ± 19.6		
receiving to root	mW	704 ± 80	936 ± 40	1552 ± 1.6	1552 ± 8	1516 ± 4	1516 ± 4
	KB/s	752.4	742.2 ± 62.9	674.3 ± 24.5	751.2	490.9 ± 11.2	468.1 ± 5.8

TABLE IV: Power consumed (as an offset from a measured base) and throughput achieved by FTP over WiFi radio in different scenarios when operating in access point mode (all phones) and ad-hoc mode (Windows Mobile-based phones only).

(or targets) for file transmissions. All tested smartphones are equipped with memory chip on which operating system files are located. This memory has small free area (up to several dozens of MBs), which can be used to store user files. In the rest of the paper, we refer to this memory as a *root memory*, or simply *root*. Using the built-in Bluetooth file transfer, we were able to receive files only into the root memory. Also, some phones feature internal flash storage in addition to the root memory (and all phones allow memory expansion with external flash). With FTP software, we could send and receive files from both root memory and internal flash, where available (for brevity, in the following we refer for the latter as *flash*).

The results for the Bluetooth-based file transfer are provided in Table III, while for the FTP software operated over WiFi in Table IV. As hinted above, the throughput measured in file transfer with Bluetooth and with WiFi is significantly lower, in some cases up to 4 times, than the one measured with *iperf*. This is the result of the latencies incurred by the operating system when accessing persistent storage. The power consumption is also reduced due to lower throughput, yet the utility of the interfaces is lower.

In general, the Android-based phones appear to consume less power while obtaining a similar or higher throughput, thus achieving better utility than Windows Mobile-based phones. These results are consistent for both Bluetooth and WiFi and over all tested phones, suggesting that Android has more energy efficient mechanisms for wireless communication and persistent storage handling than Windows Mobile. Here too, as in the experiments with *iperf*, the performance in ad-hoc mode is very similar to that of the access point network. Also, the utility of Bluetooth appears to be 3–4 times lower than WiFi.

It can be seen that in all phones there is a difference in performance between root and flash, and that the difference is much more significant in Windows Mobile than in Android, and more acute in receiving into flash than in sending from flash. A more detailed discussion of this issue is beyond the scope of this paper. Briefly, this issue indicates differences between the way these operating systems treat persistent storage of different types, and caching policies they employ for it.

Notice that the power consumption of Diamond’s WiFi is always very high. We believe it is related to a relatively poor

power management employed in Diamond, in which the power cost of WiFi depends only slightly on the provided throughput, while the base cost for holding the radio in sending or receiving state is very high. This claim is validated in experiments we made with *iperf* and UDP. These results (dropped for lack of space) show that the power consumption of Diamond’s WiFi radio even for very low throughput is very high.

V. DISCUSSION

A. Selecting an Interface for a File Transfer

When multiple interfaces are available at the same device, one would like to know which interface is more energy-efficient for a given communication pattern. The power-throughput tradeoffs discussed above can answer this question.

Specifically, suppose we need to transfer data consisting of N bytes. Denote the power consumed by interface i when sending data as P_i^{send} , while the power consumed when idle as P_i^{idle} . Also, denote the throughput of interface i as T_i . The energy wasted by interface i to send the file, E_i^{send} , is given by

$$E_i^{send} = P_i^{send} \cdot t_{send} = P_i^{send} \cdot \frac{N}{T_i} = \frac{P_i^{send}}{T_i} \cdot N, \quad (6)$$

assuming all other interfaces are switched off.

Consequently, the interface that should be selected for the transfer is the one that has a minimal $\frac{P_i^{send}}{T_i}$ ratio. Following the measurements of power and throughput presented in Tables III and IV, it is clear that for all tested smartphones, WiFi is always preferable regardless of the file size.

In a common scenario of mobile networking, a device does not switch an idle interface completely off, but rather puts it into sleep mode if possible (e.g., PSM mode of WiFi), or leaves it idle otherwise. In this case, the selected interface i should be the one that minimizes the following function:

$$E_i^{send} + \sum_{j \neq i} E_j^{idle} = \frac{P_i^{send} + \sum_{j \neq i} P_j^{idle}}{T_i} \cdot N \quad (7)$$

Here too, the energy-efficient way to transfer a file of any size from any tested smartphone would be to use the WiFi interface.

Notice that in our simplified analysis, we did not consider energy costs required to switch from idle state to sleep and back from sleep to idle. The works that did analyze these costs for various WiFi cards [19], [20] report negligible numbers

for the former transition and numbers varying from 0.2 to 1.3 Joule for the latter. Thus, even assuming that Bluetooth has zero transition costs, it can become attractive only when a few bytes are to be sent. Consequently, WiFi remains a preferable choice virtually for any interesting file size.

B. Ubiquitous Ad-Hoc Networking

As mentioned in the Introduction, WiFi enabled smartphones are becoming extremely popular. This poses a great potential for smartphone-based ubiquitous mobile ad-hoc networking, including new social and collaborative applications that may utilize physical proximity between communicating devices (see, for example, Apple's iGroups and Nokia's Instant Community). The results presented in Table I expose two main obstacles for this mode of networking. The first and more significant is the prohibitively high power consumption of WiFi when connected to an ad-hoc network. The second obstacle is the power consumption when searching for available networks.

The first obstacle sharpens the need for a distributed implementation of a PSM for ad-hoc networks. Distributed PSM appears in the standard of IEEE 802.11 [17] with additional efforts to enhance the standard implementation [21]–[23]. Yet, as our measurements show, PSM is not implemented by the phone's WiFi card driver when connected to an ad-hoc network.

The second obstacle is partially addressed in recent works on WiFi availability prediction [7], [24]. The idea is to avoid searching for available networks in places that are known to be lacking an access point coverage, based on location information or other available radios (e.g., Bluetooth [7]). This approach, however, is inappropriate for ad-hoc networks, which can be created virtually anywhere and anytime. Moreover, due to high power costs of Bluetooth scanning (see Table I), the applicability of approaches like [7] is doubtful. Thus, alternative solutions are required, tailored to the setting of ad-hoc networks.

C. Case for a Cross-Layer Throughput-to-Power Optimization

As we have seen in Section IV-B, whenever a non-negligible number of messages are dropped at the MAC layer, but their signal is sensed at the receiver, the device experiences a surge in the power consumption of the WiFi card. This is because the card needs to spend energy on receiving many messages that it either cannot successfully decode, or that it is going to drop. Unfortunately, using TCP's default settings, it will often yield a maximal throughput that is wasteful in terms of power-to-throughput, as elaborated in Section IV-B2. We have proposed a manual way to set TCP's receive buffer size such that it stops TCP flow at the maximal throughput that is also efficient in terms of power, and have demonstrated that indeed it works.

Yet, this setting depends on the specific hardware of each device and the load in the network. Hence, there is a need for a dynamic mechanism that would self adapt. Such a mechanism could be developed based on information about dropped packets and failed receptions from the network card. Implementing and tuning this mechanism is left for future work.

ACKNOWLEDGEMENT

We would like to thank Moshe Namer, the head of the Communication Laboratory in the Department of Electrical Engineering, Technion, for providing us with the equipment for performing the experiments. Special thanks are to Yuri Komorovsky for the help with setting up the environment.

REFERENCES

- [1] T. Pering, Y. Agarwal, R. Gupta, and C. Power, "Coolspots: Reducing the power consumption of wireless mobile devices with multiple radio interfaces," in *Proc. ACM MobiSys*, 2006, pp. 220–232.
- [2] P. Bonnet, A. Beaufour, M. B. Dydenborg, and M. Leopold, "Bluetooth-based sensor networks," *SIGMOD Rec.*, vol. 32, no. 4, pp. 35–40, 2003.
- [3] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Trafficsense: Rich monitoring of road and traffic conditions using mobile smartphones," Microsoft, Technical Report MSR-TR-2008-59, 2008.
- [4] J.-C. Cano, J.-M. Cano, E. Gonzalez, C. Calafate, and P. Manzoni, "Power characterization of a bluetooth-based wireless node for ubiquitous computing," *Proc. IEEE ICWMC*, p. 13, 2006.
- [5] R. Friedman and A. Kogan, "Efficient power utilization in multi-radio wireless ad hoc networks," in *Proc. OPODIS*, 2009, pp. 159–173.
- [6] —, "Power aware management middleware for multiple radio interfaces," in *Proc. 10th Int. Middleware Conference*, 2009, pp. 288–307.
- [7] G. Ananthanarayanan and I. Stoica, "Blue-Fi: enhancing Wi-Fi performance using bluetooth signals," in *Proc. MobiSys*, 2009, pp. 249–262.
- [8] E. Shih, P. Bahl, and M. J. Sinclair, "Wake on wireless: an event driven energy saving strategy for battery operated devices," in *Proc. ACM MOBICOM*, 2002, pp. 160–171.
- [9] L. M. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," in *Proc. IEEE INFOCOM*, 2001, pp. 1548–1557.
- [10] P. Bahl, A. Adya, J. Padhye, and A. Walman, "Reconsidering wireless systems with multiple radios," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 5, pp. 39–46, 2004.
- [11] Y. Xiao, P. Savolainen, A. Karppanen, M. Siekkinen, and A. Yla-Jaaski, "Practical power modeling of data transmission over 802.11g for wireless applications," in *Proc. 1st Int. Conf. on Energy-Efficient Computing and Networking (e-Energy)*, 2010, pp. 75–84.
- [12] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proc. of the 2010 USENIX Technical Conference*, 2010.
- [13] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proc. ACM IMC*, 2009, pp. 280–293.
- [14] R. Kravets and P. Krishnan, "Power management techniques for mobile communication," in *Proc. ACM MOBICOM*, 1998, pp. 157–168.
- [15] NLANR/DAST, "Iperf," Available at <http://sourceforge.net/projects/iperf>.
- [16] A. Kogan, "On porting iperf to Windows Mobile and adding Bluetooth support," Computer Science, Technion, Tech. Report CS-2010-04, 2010.
- [17] IEEE 802.11 standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, 1997.
- [18] MSDN, "TCP/IPv4 Configurable Registry Settings," Available at <http://msdn.microsoft.com/en-us/library/ms884977.aspx>.
- [19] G. Anastasi, M. Conti, E. Gregori, and A. Passarella, "802.11 power-saving mode for mobile computing in Wi-Fi hotspots: limitations, enhancements and open issues," *Wireless Netw.*, vol. 14, pp. 745–768, 2008.
- [20] C. Sengul, A. F. H. III, and R. Kravets, "Reconsidering power management," in *Proc. IEEE BROADNETS*, 2007, pp. 799–808.
- [21] E.-S. Jung and N. H. Vaidya, "Improving IEEE 802.11 power saving mechanism," *Wireless Netw.*, vol. 14, no. 3, pp. 375–391, 2008.
- [22] S. Zou, H. Wu, and S. Cheng, "Adaptive power saving mechanisms for DCF in IEEE 802.11," *Mobile Networks and Applications*, vol. 10, no. 5, pp. 763–770, 2005.
- [23] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh, "Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks," *Comput. Netw.*, vol. 43, no. 3, pp. 317–337, 2003.
- [24] H. Wu, K. Tan, J. Liu, and Y. Zhang, "Footprint: cellular assisted Wi-Fi AP discovery on mobile phones for energy saving," in *Proc. ACM WINTECH*, 2009, pp. 67–76.