RT
00633
c.1

Energy-Aware Adaptive Compression for Mobile Devices

by

Rakan Maddah

B.S., Computer Science, LAU, 2007

Thesis submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science

Department of Computer Science and Mathematics

LEBANESE AMERICAN UNIVERSITY

July 2009

LEBANESE AMERICAN UNIVERSITY

Thesis Approval Form

Student Name	: Rakan S. Maddah I.D.: 200301926
Thesis Title	: Energy-Aware Adaptive Compression for Mobile Devices
Program	: M.S. in Computer Science
Department	: Computer Science and Mathematics
School	: Arts and Sciences - Beirut

Approved/Signed by:

Thesis Advisor	Dr. Sanaa Sharafeddine	[redacted]
Member	Dr. Ramzi Haraty	[redacted]
Member	Dr. Zaher Dawy	

Date: 30-07-2009

Plagiarism Policy Compliance Statement

I certify that I have read and understood LAUs Plagiarism Policy. I understand that failure to comply with this Policy can lead to academic and disciplinary actions against me. This work is substantially my own, and to the extent that any part of this work is not my own I have indicated that by acknowledging its sources.

Date: 30-07-2009

Name: Rakan S. Maddah

[redacted]

To My Parents

Abstract

A single bit transmission over the wireless card can consume 1000 times more energy as compared to a 32-bit CPU computation. This fact is very critical for devices operating on limited battery such as mobile devices. By applying data compression, the intended information is sent with lower number of bits and thus reducing transmission energy. However, the computational as well as memory access requirements of the compression algorithm used could consume more energy than simply transmitting data uncompressed. Moreover, if the transmission rate over the wireless medium is high then the need for compression might be reduced or even eliminated as data is transferred within a minimal amount of time. In light of this, the compression option is investigated through conducting experimental work in different scenarios to record the energy consumption of data transfer from one mobile device to another. Energy results show that compression does not lead to energy gains at all times. Whenever the received signal strength at the mobile device is high, no compression is needed. However, compression is profitable whenever the signal strength gets weak. In this thesis, an adaptive scheme is proposed that monitors the signal strength during the transmission process and compresses data on-the-fly whenever energy gain is promised otherwise sends data uncompressed. This thesis shows by means of experimental work and simulation that such an adaptive scheme results in significant gains in energy consumption as compared to other related approaches proposed in the literature. Based on the experimental results, an empirical model is finally derived to estimate the energy consumption of a given transmission.

Acknowledgment

I would like to thank my advisor Dr. Sanaa Sharafeddine for her guidance and support throughout my Thesis work. A thanks is also to Dr. Ramzi Haraty and Dr. Zaher Dawy for being on my thesis committee.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contemporary businesses are characterized by a significant changing rate of data and requirements. Such nature of businesses highlights the need for continuous and pervasive connectivity to keep track of the frequent changes. Despite their wide spread, laptop computers cannot smoothly achieve continuous connectivity as carrying a laptop is an overhead for many. Consequently, continuous connectivity shall be achieved with comfortable and reliable devices such as PDAs and smart phones. These devices have penetrated all aspects of human activities because of their convenience, cheap price, lightweight, and compact size.

Furthermore, peer-to-peer (P2P) file sharing applications such as BitTorrent [33], LimeWire [34] and Kazaa [35] have gained vast popularity. The users of these applications form among each other a P2P network [29] where a variety of files are shared and exchanged. P2P networks does not deploy the traditional client/server communication scheme. However, each participating node within the network can play the role of a server providing files for other nodes as well as a client acquiring files from other nodes. As mobile devices are available largely among people and always turned on, they seem to be suitable to form P2P network allowing effective files sharing without the need to pass through a dedicated server.

As a matter of fact, wireless networks are almost ubiquitous, so a mobile device, incorporated with a wireless card, can easily achieve continuous connectivity and share

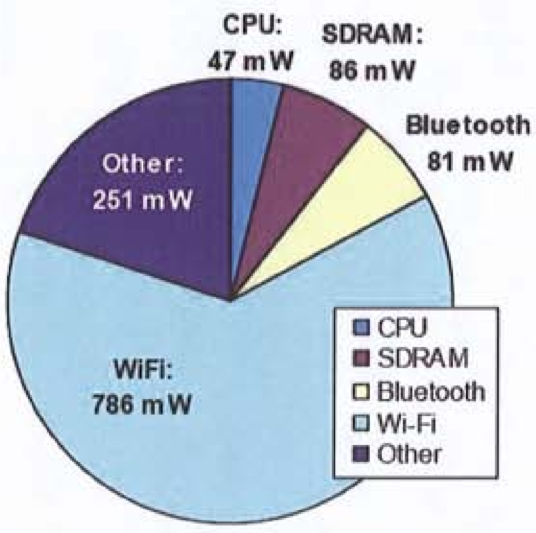


Figure 1.1: Power consumption of various components of a mobile device [2]

files. However, the limited lifetime of mobile devices' batteries is a notable constraint. Studies have shown that transmitting one bit over the wireless card requires 1000 times more energy than a single 32-bit computation [1]. In addition, wireless card consumes the greatest amount of energy even at the idle mode [2]. Fig. 1.1 shows that the wireless card consumes around 70% of the total power of a mobile device in idle mode. Therefore, reducing the amount of transmitted data is highly desirable for battery-constrained devices with wireless connectivity.

Schemes that promote continuous connectivity and file sharing, while reducing the amount of energy consumption are needed. To this end, compression looks to be a promising scheme. By applying data compression, the intended information is sent with a lower number of transmitted bits; thus reducing transmission energy and likely prolonging the battery lifetime. However, data compression can backfire. As a matter of fact, the computational requirements for a compression algorithm could be very high, which might lead to consume more energy than to send data uncompressed. Moreover, if the transmission rate over the wireless medium is high then the need for compression might be reduced or even eliminated as data is transferred within a minimal amount of time.

In this work, the option of compression is investigated and an adaptive compression scheme that accounts for the current transmission status and decides whether to compress data or not is proposed. The signal strength is considered as the main criterion for the compression decision. When the signal strength is low, the mobile device has to consume more energy to transmit data; thus it is more profitable to perform compression. However, the signal strength might improve during the transmission, which makes it more profitable to send data uncompressed.

This thesis is organized as follow: chapter 2 highlights the related work and shows how our approach differs from existing literature, chapter 3 presents the system model and the experimental setup initiated to conduct energy measurements of different scenarios, chapter 4 proposes and explains the energy-aware adaptive compression scheme and presents energy measurements results in order to demonstrate the achievable gains, chapter 5 demonstrates the usefulness of the adaptive approach in a multiple-hops environment and chapter 6 concludes the thesis and proposes future work.

Chapter 2

Literature Review

To stay updated, people are after achieving continuous connectivity through the convenience that mobile devices provide. Since their emergence, these devices suffered from the inherent limitation of the battery lifetime. To this end, many energy saving approaches were proposed. In [3], energy approaches are grouped under the umbrella of four main categories: computer networks, operating systems, applications design, and computer architecture.

In computer network category, many protocols have been proposed at the different internet stack lyaers to reduce energy. At the physical layer, the EC-MAC protocol was designed to reduce the number of retransmissions through avoiding collisions at the MAC layer [4]. PAMAS is another protocol that switches off the transceiver whenever it senses an idle period characterized by an absence period of data reception [5]. Furthermore, energy efficient error control such as ARQ [6] have been proposed. At the transport layer, TCP is a well known transport protocol that is used to guarantee the transfer of data without loss. However, TCP turned out not to be suitable for wireless environments as it attributes packet loss related to wireless factors to congestion. Thus, it imposes a high number of retransmissions in wireless environments, which leads to performance degradation and high energy consumption. To alleviate this problem, a variation of TCP such as I-TCP [9] and M-TCP [10] were presented to alleviate the problems of normal TCP and reduce energy consumption. At the application layer, APIs that offer functionalities to reduce energy were presented in order to be used by developers. Power Monitor [11] and Power Interface [12] are examples of such APIs.

In the latter three categories, many techniques to reduce energy have been proposed. In [14], a tasks scheduling policy for the operating systems was presented. The role of the policy was to assign the tasks in such a way that best exploits the CPU. In [13] and [15] operating systems design issues were addressed to reduce energy consumption. At the application level design, software development models such as ACPI [16] were presented to guide software design from an energy saving perspective. When it comes to the computer architecture level, many techniques to reduce the energy of the hardware were proposed. In [24], a scheme that turns the wireless card into sleep mode when it senses inactive period is presented. In [25], a technique to reduce the connection setup time for the wireless card and the discovery time for the Bluetooth that resulted in efficient energy savings is presented.

The idea of applying compression a-priori to sending on mobile devices is exploited in [17]. To compress data, a set of lossless compression algorithms is selected with which experimental work is conducted resulting in interesting observations. For the majority of the selected compression algorithms, the amount of energy consumed in sending data compressed is comparable to the uncompressed case. This is due to the fact that compression algorithms require intensive computational requirements that consume significant amount of energy. The authors in [17] proposed that data should be always compressed before transmission and the used compression algorithm should undergo code optimization to reduce the computational and the memory access requirements to render it suitable for battery-constrained devices such as mobile devices or PDAs. They also proposed to apply asymmetric compression where different compression/decompression algorithms are used based on their energy requirements. In this work, we propose an adaptive and practical compression scheme that works on-the-fly and decides for every block of data whether to be transmitted compressed or uncompressed. Our proposed approach is a practical approach that utilizes a lightweight compression algorithm without performing any code optimization.

In [18], the authors focus on the energy requirements of the decompression process at the mobile device. It is assumed that the mobile device requests that the server compresses the data using the compression algorithm that results in the highest energy savings for decompression at the mobile device side. In this work, we consider the compression operation performed at mobile devices for peer-to-peer communications.

In [19], an adaptive compression environment for personal computers is presented. The decision to compress or not to compress is based on the network status and bandwidth availability. The initial step is to estimate the current available bandwidth, and based on the results one out of three compression algorithms is selected. This adaptive compression scheme is shown by measurements to be effective in terms of increasing the network bandwidth availability. In this work, we consider battery-constrained devices rather than personal computers and thus we aim to have a simple and practical compression scheme that results in high energy savings for such devices.

2.1 Compression Algorithms

Compression could be defined as the process of representing an amount of data with less number of bits. To achieve this target, most compression principles rely on statistical techniques. These techniques detect patterns and redundancy in the data to be compressed. The role of the compression algorithm is to encode these patterns in a way that diminishes the number of bits representing the original data.

Compression is divided into two main categories. Lossy compression that does not totally preserve the original data. Some parts of the original data is lost after decompression. Lossy compression is applied to data that tolerates some loss such as audio and video. On the other hand, lossless compression perfectly regenerates the original data after decompression. Lossless compression is applied on data types that cannot tolerate loss such as textual data.

The compression algorithms explored in this thesis fall under the lossless compression category. Four lossless compression algorithms that have different computational requirements and compression ratios are selected from #ZipLib [23], which is an open shared library for the .Net Compact Framework implemented entirely in C#. These are: Zip [27], Bzip2 [26], Gzip [31], and Deflate [32]. Among these, two compression levels of Zip are used: Level 1 and Level 9. Level 1 achieves the lowest compression ratio and level 9 achieves the highest compression ratio. Except Bzip2, the other compression algorithms are based of the famous Lempel-Ziv [30] compression algorithm. The latter relies on a dictionary that contains an encoding for all possible characters. When data is to be compressed, the algorithm takes strings of input and starts processing it. For every word it encounters within the dictionary, the algorithm applies encoding. When two encodings come one after the other, the algorithm forms one encoding and adds it to the dictionary. Though Lempel-Ziv is deployed by these algorithms, each has it is own implementation. In addition to Lempel-Ziv, the algorithms apply other complementary techniques. For example, Deflate applies Huffman Coding [28], which is an encoding scheme based on representing the symbols with short encodes with the help of binary trees. As for Bzip2, it employs several compression techniques that are applied in sequence such as run-length encoding that encodes any repeat of four characters or more by one character followed by the number of repeats.

Chapter 3

System Model

Applying compression ahead of sending data seems to be a valid scheme to reduce energy consumption for mobile devices, while transmitting over the wireless card. However, the majority of the compression algorithms require intensive computational power. Such requirements could maximize the rate of battery drain instead of minimizing it. To assess the profitability of applying compression, energy measurements were recorded through conducting experimental work throughout the thesis. The experiments consisted of connecting two mobile devices through a TCP connection while one plays the role of a sender that transmits a file to the other receiving device. TCP was deployed as it guarantees the transmission of the file without any losses. To transmit data, the file is split into blocks of data. Each block was sent either compressed or uncompressed depending on the experimental scenario requirements. For some experiments, only two devices were engaged. The energy consumption of the transmitting device was always recorded. For other experiments, additional nodes were introduced to play the role of relays between a source and a destination. For such a setting, the energy consumption for all the transmitting nodes was recorded.

3.1 Experimental Setup

To conduct experimental work, the following experimental setup was initiated.Two HP iPAQ hx2400 series were used with the following specifications [20]: Microsoft Windows Mobile 5.0, 64 MB of RAM, 520 MHz Marvell PXA270 processor and an integrated 802.11b wireless card. To measure the energy consumption of sending data, a typical energy measurement technique was used to measure the voltage drop across a known resistor that connects serially the positive pin of the battery to that of the device itself [22]. The measurements are then used to calculate the static power drain of the battery. The measurement setup, shown in Fig. 3.1, uses a data acquisition unit (DAQ USB-6251 [21]) fabricated by National instruments to capture the voltage fluctuations across the resistor.

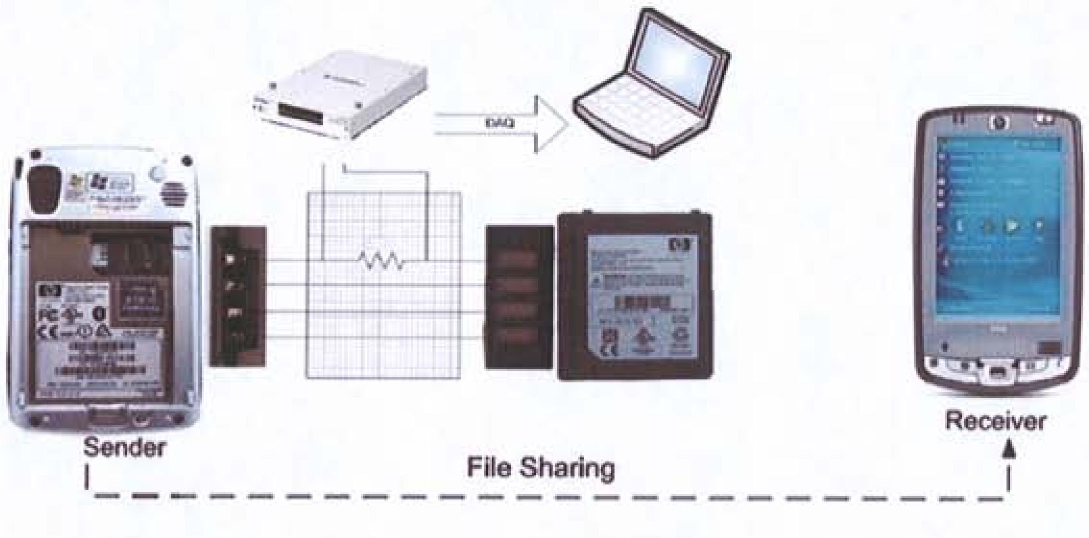


Figure 3.1: Experimental setup

The two devices were connected wirelessly, relying on the wireless card medium, in ad-hoc mode through a TCP connection. To apply compression, the Zip, Deflate, Gzip and Bzip2 compression algorithms presented earlier were used. Experimenting with various compression algorithms of various computational requirements is essential to assess the profitability of whether data compression should be applied or not.

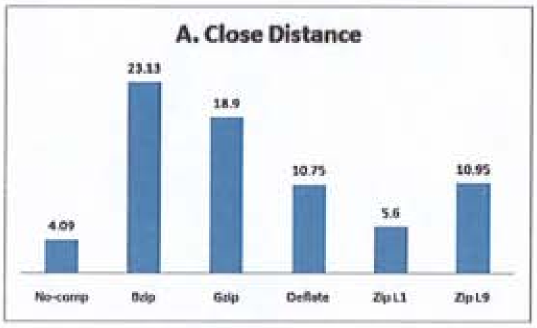
3.2 Problem Formulation

As stated previously, the aim behind conducting experimental work is to assess the profitability of applying data compression. Subsequently, The energy required to send data with and without compression should be monitored. Accordingly, the experiment work was carried along two different schemes. The no-compress scheme is the prevailing approach of sending data without applying data compression. The always-compress scheme is the approach of compressing the data a-priori to sending as was proposed in [17]. The always-compress scheme was implemented with the various compression algorithms presented previously.

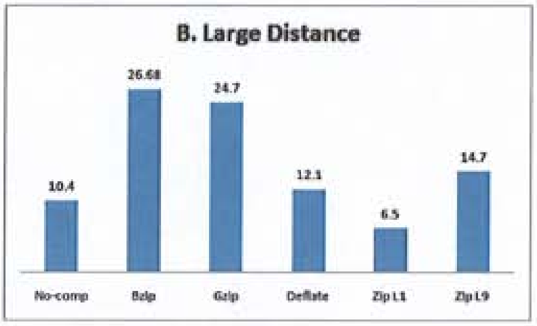
To compare energy consumption for the no-compress and always-compress schemes, several measurements were conducted with two settings. The first setting consists of placing the two devices close to each other (1 m separation) and the second consists of placing them distant from each other (12 m separation). These settings are selected to provide different channel conditions where a strong signal is achieved when the devices are close in distance and a weak signal is achieved when the devices are distant. A 2.52 MB text file was used to be transferred from one device to the other.

Fig. 3.2 presents a comparison of energy consumption between the always-compress and the no-compress schemes. Fig. 3.2 (a) shows that the no-compress approach is the most energy-effective approach (i.e., the one that consumes the least amount of energy) when compared to the always-compress approach implemented with the various compression algorithms. These results require the investigation of the compression capabilities for the considered algorithms.

Table 3.1 shows that Bzip2 achieved the highest compression ratio (75%). However, it takes a relatively long processing time to finish (41 s). Gzip and Deflate exhibited a comparable compression ratio but finish in twenty one seconds less than Bzip. Zip Level 9 achieved close compression results in comparison with Gzip and Deflate, but it takes three more seconds than the latter two to finish. Zip Level 1 has the lowest compression ratio but is remarkably fast. It is interesting to notice that Gzip and Deflate



(a)



(b)

Figure 3.2: Energy measurements in Joule for a 2.52 MB text file

achieved the same compression ratio, but Deflate requires less energy. Furthermore, Zip L9 achieved good energy results though it takes more time to compress than Gzip and Deflate. These results highlights the influence of the computational requirements that compression algorithms necessitate.

Table 3.1: Comparison of compression algorithms

Compression Scheme	File Size	Compression Ratio	Compression Time
No Compression	2.52 MB	0%	-
Bzip	626 KB	75%	41 s
Gzip	780 KB	69%	20 s
Deflate	780 KB	69%	21 s
Zip Level 1	984 KB	60%	7 s
Zip Level 9	799 KB	68%	23 s

From the measurements of Fig. 3.2 (a), we observe that compression is not a good option when the devices are placed close to each other. Eventhough less bits are being transmitted, compressing the file a-priori to sending did not yield in any energy saving with the four different compression algorithms . At a close distance, the devices acquire strong signal strength improving their energy consumption for wireless transmission because data is transmitted at a high bit rate and the number of retransmissions and errors is little. Thus, the overhead of CPU computations and memory access imposed by the compression process require more energy than sending the data uncompressed. As a result, it is more energy-profitable to send the data uncompressed rather than compressed along a strong signal strength.

Therefore, it is interesting to investigate the compression option in the case of a weak signal strength. Fig. 3.2 (b) shows the energy consumption results when the communicating devices are placed distant from each other. With a weak signal, the no-compress approach is no more the most energy-effective approach; its energy consumption has increased remarkably from 4.09 to 10.4 Joule. Despite this increase, the compression algorithms do still perform worse or similar to the no-compress approach except for Zip Level 1. The latter achieved 37.5% of energy gain as compared to no-compress scheme.

From the results of Fig. 3.2 (b), interesting conclusions can be derived . Once again, the computational requirements for the majority of the compression algorithms consume more energy than sending the data uncompressed. Zip Level 1 characterized by light computational requirements and fast compression time, lead the always-compress scheme to achieve energy gains over the no-compress scheme. Thus, compression can be an energy saving scheme whenever the signal strength of the communicating devices is weak and the deployed compression algorithm is light in terms of computational requirements.

The above results showed that compression leads to energy saving only with a weak signal strength. In addition, the energy consumptions imposed by the computational requirements of the compression algorithms make compression unprofitable with a strong signal. This observation initiates the possible introduction of a new compression scheme that should not apply compression except when energy gains are promising.

Chapter 4

Energy-Aware Adaptive Compression Scheme

The observations noted in chapter 3 motivate the development of an adaptive compression approach. With a strong signal strength, compression should not be an option as it consumes more energy than sending data uncompressed. On the other hand, compression using computationally light algorithms is energy-profitable whenever the signal strength is weak.

Based on these observations, an adaptive on-the-fly compression scheme for mobile devices that monitors the signal strength and decides accordingly to apply compression or not is proposed. For this scheme, data is to be read into blocks during the transmission and some blocks may be sent compressed and others uncompressed depending on the signal strength.

This adaptive scheme was implemented and its work flow is presented in Fig. 4.1. The scheme starts by reading one block of the data to be transmitted and checks the signal strength. A signal that goes above a certain threshold T is considered strong, otherwise weak. The adaptive scheme proceeds by sending one block of data uncompressed if the signal strength is strong and compressed using Zip Level 1 if the signal strength is weak.

The proposed adaptive compression scheme combines the best out of no-compress and always-compress schemes. It applies compression only when energy reduction is promising and discard it when not. Furthermore, the adaptive approach is characterized by its simplicity which makes it suitable for mobile devices. The latter have limited resources that does not fit complexity. The adaptive compression scheme has a linear run time in order of O(n), where n is the number of blocks to be transmitted, in addition to the run time of the compression algorithm.

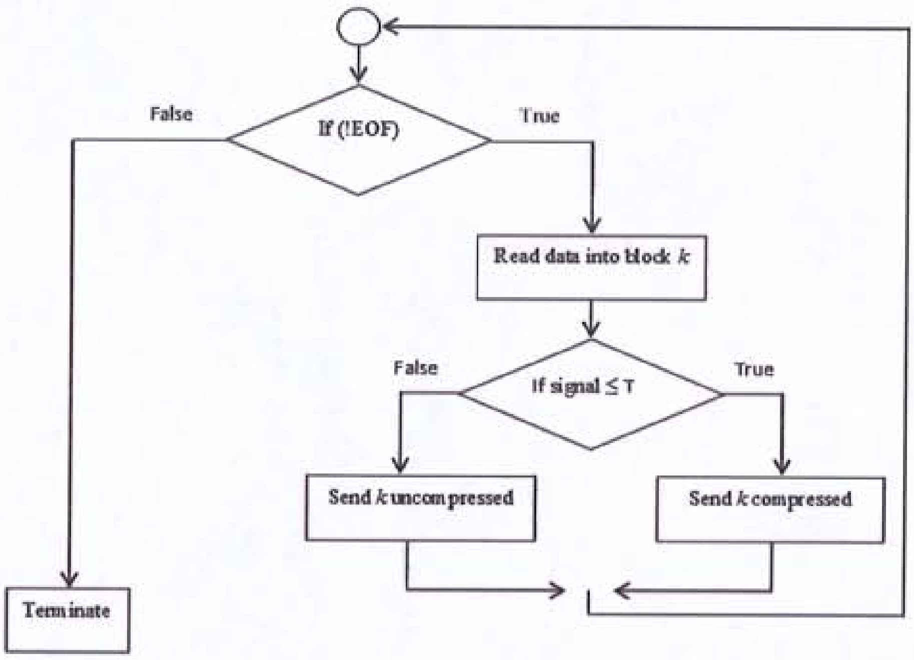


Figure 4.1: Adaptive scheme flowchart

4.1 Signal Strength Threshold Parameter

The proposed adaptive compression scheme applies compression only when the signal strength is below a certain threshold. Setting an appropriate threshold is essential to get the best out of the adaptive scheme in terms of energy saving. To determine this threshold, an experimental work was carried. A 5.05 MB text file was sent between the two PDAs along two different scenarios. The first scenario consists of sending the entire file compressed while the second scenario consists of sending the entire file uncompressed. The experiment was conducted by starting the sending process with a strong signal and ending in a weak signal. For each block of sent data, the signal strength at which the block was sent is recorded and the energy consumed per block basis is calculated. It should be noted that for this experiment a large file size was used because of the need to prolong the transmission time to acquire a variety of signal strength values. Furthermore, the file was split into a specific number of blocks of data because of the need to pause the transmission after each block to be able to identify the energy per block. A limited number of block is required as the energy measurements techniques adopted can record energy for a limited time period.

An analysis of the energy results presented in the previous chapter would reveal that at certain signal strength value, the energy of sending data uncompressed should exceed the energy of sending data compressed. This deduction manifested itself in the experimental work results as shown in figure Fig. 4.2.

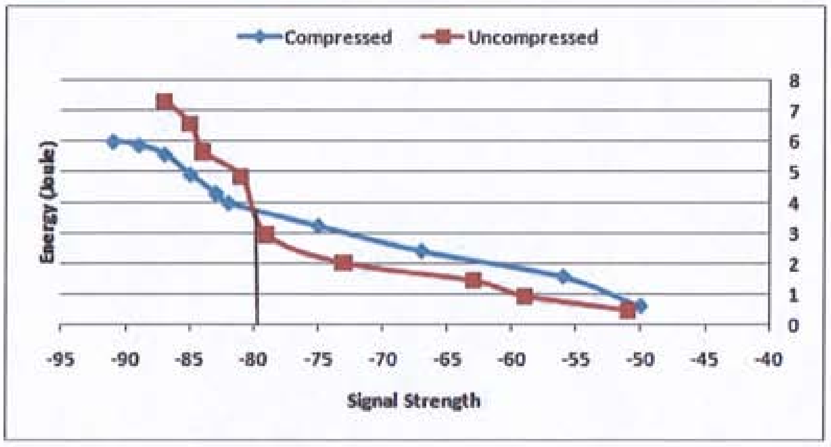


Figure 4.2: Signal strength threshold for compression

From Fig. 4.2, it can be seen that -79 decibel is to be set as the signal strength threshold for compression. The energy of sending data uncompressed is consuming less energy than sending it compressed when the signal is greater than -79. However, when the signal strength attains a value less than -79, the energy of sending data uncompressed is greater than sending data compressed.

4.2 Adaptive Compression Scheme Assessment

To assess the performance of the proposed adaptive approach, energy measurements were conducted with four different data block sizes: 4 KB, 8 KB, 12 KB and 16 KB. The variability of data block sizes was to investigate the influence of the data block size on energy consumption. For each block size, the experiment was conducted along three different scenarios. The first scenario consists of achieving a strong signal strength by placing the devices next to each other (less than 1 m) in order to send the entire file without applying compression. The second scenario consists of alternating the signal strength from strong to weak in order to send the file with compressed and uncompressed blocks. The third scenario consists of achieving weak signal through placing the devices distant from each other (10 m) in order to send the entire file compressed. The alternating signal is achieved by varying the distance between the devices through starting by placing the devices next to each other, then increasing the distance subsequently by moving the receiving device to attain a weak signal. As a matter of fact, the signal strength is highly affected by the distance. Fig. 4.3 shows several signal strength values recorded at different distances. The plot indicates that the signal strength weakens remarkably with the increase in distance. The experiment was carried with the same text file used with the always-compress and the no-compress schemes.

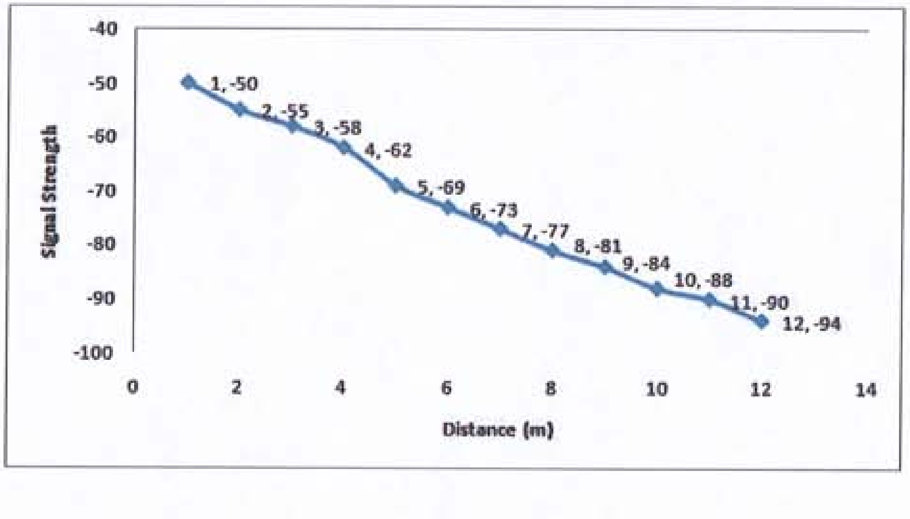
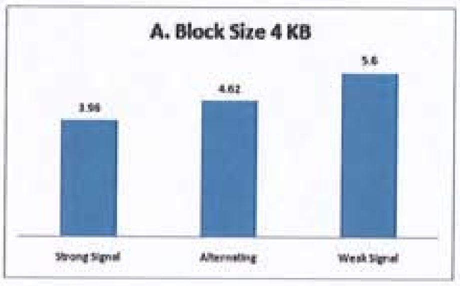
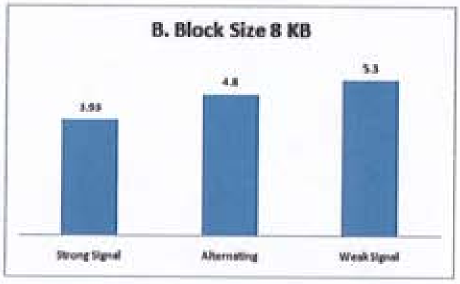


Figure 4.3: Signal strength variation as a function of distance

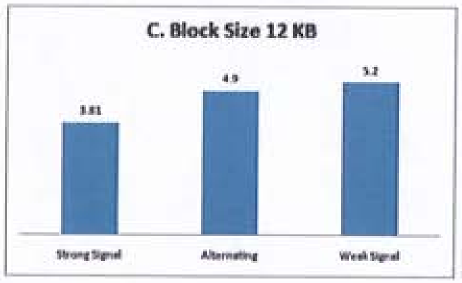
The energy results shown in Fig. 4.4 demonstrate the fruitfulness of deploying the adaptive compression scheme. To better evaluate these results, they should be compared to the results of the no-compress and the always-compress schemes presented in the previous chapter. In case of the no-compress scheme along a weak signal strength, 10.4 Joule are needed to send the file (refer to Fig. 3.2 (b)). However, the adaptive scheme requires 5.07 Joule when the block size is 16 KB (refer to Fig. 4.4 (d)). Thus, significant energy improvements that amount to 51.25% are achieved with the adaptive scheme. In case of always-compress scheme along a strong signal strength, 5.6 Joule are needed to send the file compressed via Zip Level 1 (refer to Fig. 3.2 (a)). However, the adaptive scheme sends the data with 3.7 Joule (refer to Fig. 4.4(d)) with a block sizes of 16 KB achieving 34% saving in energy. The remarkable energy gain that the adaptive compression scheme introduces demonstrates its superiority to other approaches. The strength of the proposed approaches comes from its ability to adapt to the condition of the wireless communication channel indicated by the signal strength.



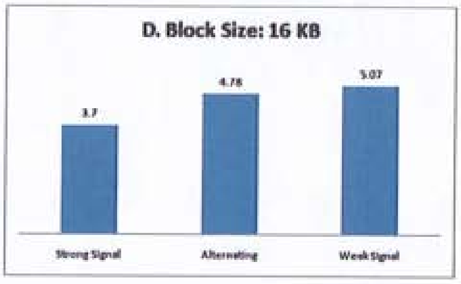
(a)



(b)



(c)



(d)

Figure 4.4: Adaptive scheme energy measurements in Joule.

4.2.1 Results Interpretation

The results of Fig. 4.4 incorporate interesting observations that should be noted. The always-compress scheme required 6.5 joule (refer to Fig. 3.2 (b)) when the signal is weak. However, the adaptive approach requires 5.07 Joule (refer to Fig. 4.4 (d)) with the same signal. Accordingly, the adaptive approach beats the always-compress approach by 22% of energy gain. Though with a weak signal both schemes compress data, the adaptive scheme applies compression on the fly. As a matter of fact, the always-compress scheme requires compressing the file a-priori to sending. Meanwhile, the wireless card is turned on and idle causing additional energy consumption. By applying compression on-the-fly, the adaptive scheme almost eliminates the idle duration time of the wireless card, which results in energy saving.

Furthermore, the energy required to send the file with strong or weak signal strength is decreasing with the increase of the block size. Actually, increasing the block size implies less exploitation of the adaptive scheme as the signal strength is checked less, i.e. the signal strength is checked three times less for a block size of 16 KB in comparison to a block size of 4 KB. As a consequence to this observation, one can deduce that a large block size should be adopted whenever the signal strength maintains a steady state (strong or weak). It is thus preferable to adopt a small block size when the signal strength is alternating and a larger block size when the signal strength is steady. If the signal strength is checked less, then less computational energy is required and less idle time duration for the wireless card is achieved. However, checking the signal more frequently when the signal alternates is preferable as the adaptive scheme is more exploited implying the application of compression when promising.

4.2.2 Influence of the Block Size on Energy Consumption

From the pervious energy results, it was claimed that a large block size should be adopted when the signal strength is steady (strong or weak). A large block size would cut down on the number of checks for the signal strength, which shall lead to energy savings as less computations are achieved. To validate the claim, an experimental work was carried to record the energy results of the adaptive scheme with a block size of 1 KB and 64 KB. The experiment was carried with the same text file as the one used to get the energy results of Fig. 4.4. The energy results for the adaptive scheme with a block size of 1 KB and 64 KB are shown in Fig. 4.5.

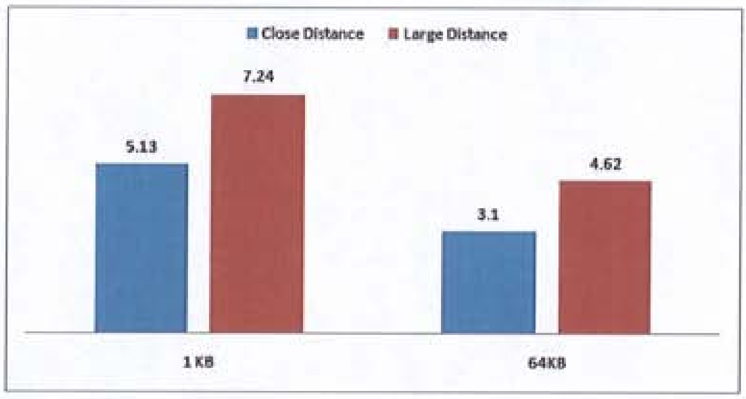


Figure 4.5: Adaptive scheme with 1 KB and 64 KB blocks energy results

The results show that with a block of 64 KB, the adaptive scheme scored 39.57% and 36.04% of energy gain with a strong and weak signal respectively over a block of size 1 KB. The achieved gain validates the claim of adopting a large block size with steady signal.

4.2.3 Idle Time Evaluation

The energy results presented so far have shown that the adaptive compression scheme scores better energy results when compared to the always-compress scheme even with a weak signal. The justification for this result was that the adaptive scheme diminishes the idle time for the wireless card, which leads to energy saving. To validate this claim, an experimental work was carried. A 3.00 MB text file was transferred between the two PDAs along a weak signal strength. A different text file was adopted to make sure that the results are independent of the file to be sent. The experiment was carried twice. The first time the file was sent with the always-compress scheme while recording the energy consumption, the compression time and the transmission time. The second time, the file was sent with the adaptive scheme while recording the energy consumption and the compression and transmission time. It should be noted that with the adaptive scheme the compression and transmission time cannot be split as data is compressed on the fly.

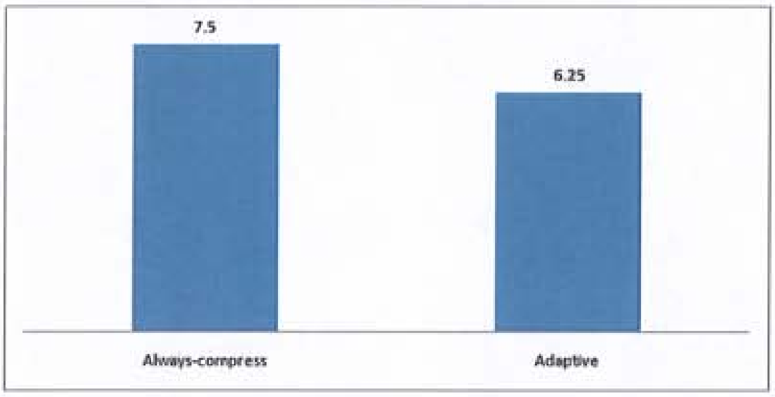


Figure 4.6: Always-compress and adaptive schemes energy results

The energy results for the conducted experiment are presented in Fig. 4.6. The adaptive scheme achieved 16.66% of energy gain over the always-compress. To justify the gain, the recorded times should be checked. The always-compress scheme took 7 seconds to compress the file and 8 seconds to transmit the data. On the other hand, the transmission and compression time for the adaptive scheme took 11 seconds. Thus, the adaptive scheme accomplished the same mission in 4 seconds less. This shows that the adaptive scheme cuts down on the idle time, which makes the adaptive scheme scores better energy results than the always-compress scheme even with a weak signal.

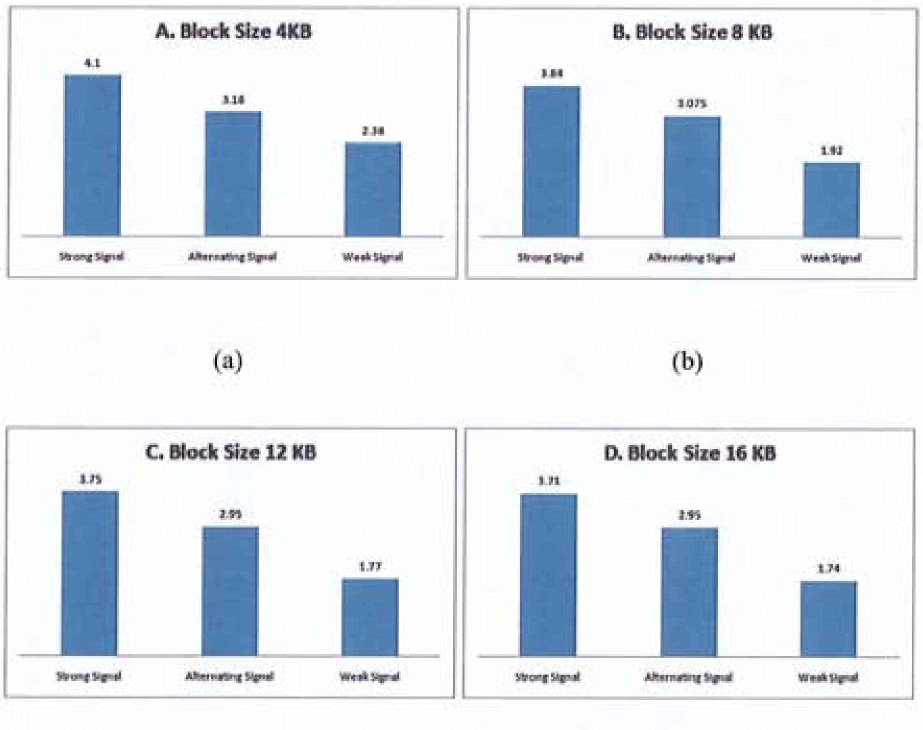
4.3 Performance Evaluation as a Function of the File Compression Ratio

To further validate the proposed adaptive compression scheme, an additional experimental work was conducted. The experiment was carried with 4 different generated binary data files of size 2.92MB each. The first file formed of 100% of zeros, the second file is formed of 50% of zeros and 50% of ones, the third file is formed with 10%

of zeros and 90% of ones and the fourth file is formed with 25% of zeros and 75% of ones. The aim behind experimenting with the benchmark of binary files is to assess the fruitfulness of the adaptive scheme when faced with data that achieve various compression ratios. For each file, the energy consumption for the adaptive, the no-compress and the always-compress schemes were collected.

4.3.1 All-Zeros File

The results of the energy consumption while sending the 2.92 MB all-zeros binary file are presented in this section. Since this file is made of zeros only, then its compression process is easy and fast achieving a compression ratio of 99.54%. The compression algorithm has to account for few entries within its compression dictionary.

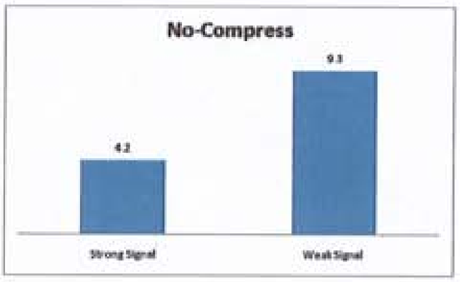


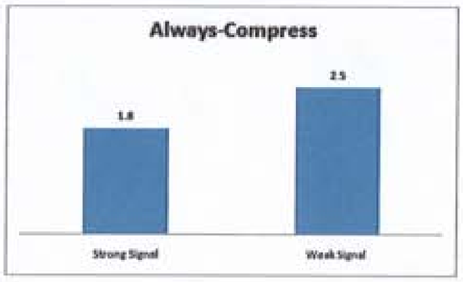
(a) (b)

(c) (d)

Figure 4.7: Adaptive scheme all-zeros binary file energy results

The results of Fig. 4.7 and Fig. 4.8 show that with a weak signal, the adaptive approach requires 1.74 Joule (refer to Fig. 4.7 (d)) while the no-compress approach





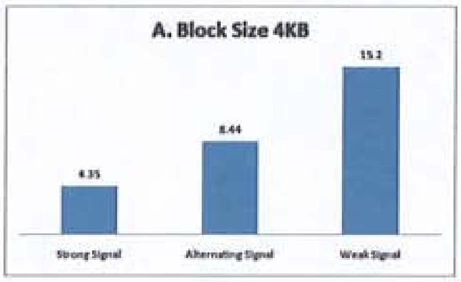
(a) (b)

Figure 4.8: All-Zeros binary file energy results

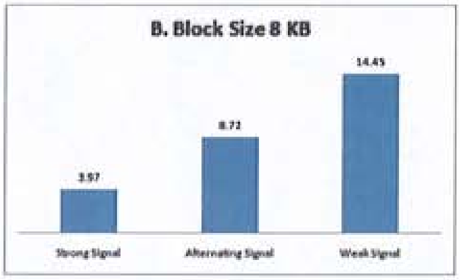
requires 9.3 Joule (refer toFig. 4.8 (a)). Thus, the adaptive approach scores-up to 81.29% of gain in energy as compared to the no-compress scheme. In comparison to the always-compress approach, the latter beats the adaptive scheme with a strong signal. This is due to the fact that the all zeros binary file is easily compressed and does not require high computation power. Nevertheless, the adaptive scheme (1.74 Joule) beats the always-compress scheme (2.5 Joule) when the signal is weak as it sends data compressed on the fly. Furthermore, with an alternating signal the adaptive approach requires 2.95 Joule (refer to Fig. 4.7 (d)). If we consider the average of the energy consumption with a strong and weak signal for the always-compress approach as an indicator of the energy consumption for an alternating signal, then the always-compress approach requires 2.15 Joule (refer to Fig. 4.8 (b)). Thus, the adaptive and always-compress approaches act closely with an alternating signal. Based on these results, it can be deduced that the adaptive scheme performs similarly to the always-compress scheme when applied on data file that has the characteristics of the all-zeros file. However, the adaptive approach extensively beats the no-compress approach with such type of files.

4.3.2 50-50 File

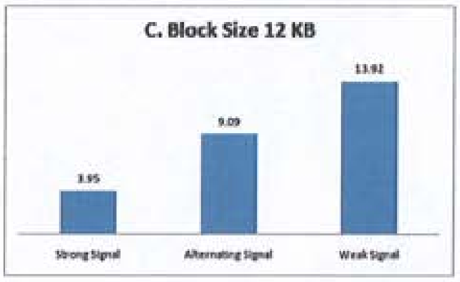
A 50-50 binary file is characterized by its inability for being compressed. It achieves a compression ratio of -1% because of the space requirements for the compression algorithm variables.



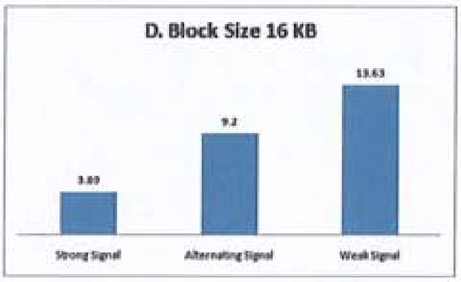
(a)



(b)

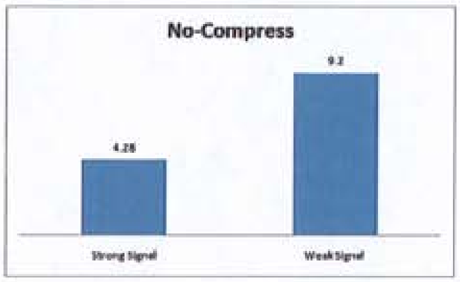


(c)

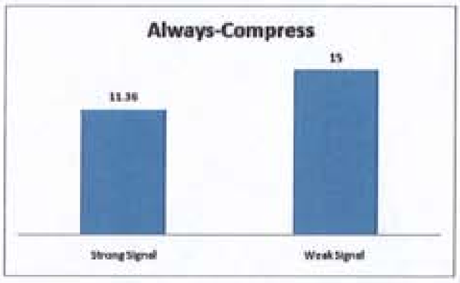


(d)

Figure 4.9: Adaptive scheme 50-50 binary file energy results



(a)



(b)

Figure 4.10: 50-50 binary file energy results

The results of Fig. 4.9 and Fig. 4.10 show that when the signal is strong, the adaptive scheme can send the data with 3.89 Joule (refer to Fig. 4.9 (d)) while the always-compress scheme requires 11.36 Joule (refer to Fig. 4.10 (b)). Thus, the adaptive approach beats the always-compress scheme by a gain in energy up to 65.75% with a strong signal. However, the no-compress scheme (9.2 Joule) beats the adaptive scheme (13.63 Joule) by 31.13% when the signal is weak. This difference in energy consumption is due to the fact that the 50-50 binary file has a compression ratio less than zero. Nevertheless, such a result can be easily avoided by making the adaptive approach skip over compression when the file to be sent is uncompressible. These files have well known extensions such as PDF, MP3, MPEG etc.

4.3.3 10-90 File

The energy consumption results to send the 2.92 MB 10-90 binary file are presented in this section. The 10-90 binary file achieves a compression ratio of 41.78%.

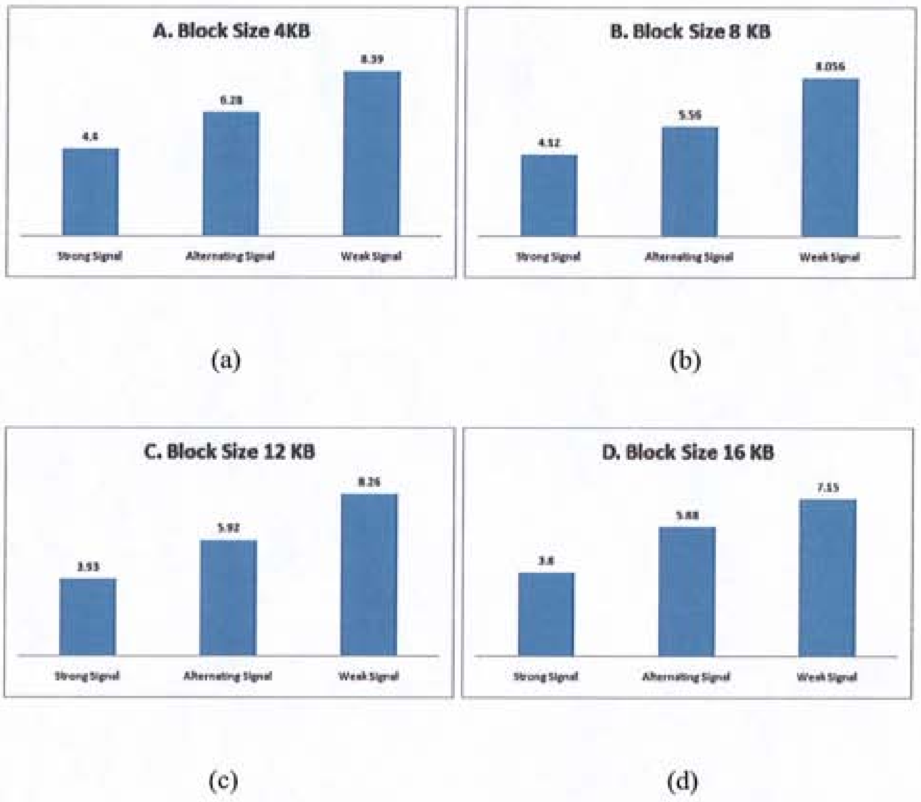
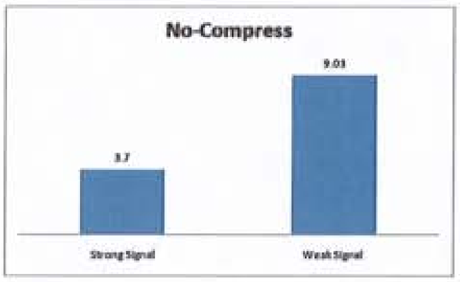
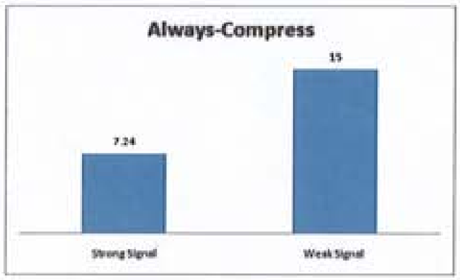


Figure 4.11: Adaptive scheme 10-90 binary file energy results

The results of Fig. 4.11 and Fig. 4.12 show that when the signal is strong, the adaptive scheme can send data with 3.8 Joule (refer to Fig. 4.11 (d)) while the always-compress scheme requires 7.24 Joule (refer to Fig. 4.12 (b)). Thus, the adaptive ap-



(a)



(b)

Figure 4.12: 10-90 binary file energy results

proach beats the always-compress scheme by up to 47.51% of energy gain with a strong signal. Furthermore, the adaptive scheme (7.15 Joule) beats the no-compress scheme (9.03 Joule) by 20.48% with a weak signal. In conclusion, the results show that the adaptive scheme beats the always-compress and the no-compress schemes in all circumstances with files that have the characteristics of the 10-90 file.

4.3.4 25-75 File

This section presents the results of the energy consumption while sending the 2.92 MB 25-75 binary file. This binary file achieves a compression ratio of 14% and is hardly compressed.

The results of this file, as shown in Fig. 4.13 and Fig. 4.14, and their interpretation are very similar to the 50-50 file, but with no possibility of making the adaptive scheme skip over compression for such files. The data file is hard to compress, which requires intensive computational requirements imposed by the compression algorithm. These requirements makes it more preferable to discard compression in all scenarios. Thus, it can be deduced that the no-compress scheme beats the adaptive scheme with such files only when the signal strength is weak.

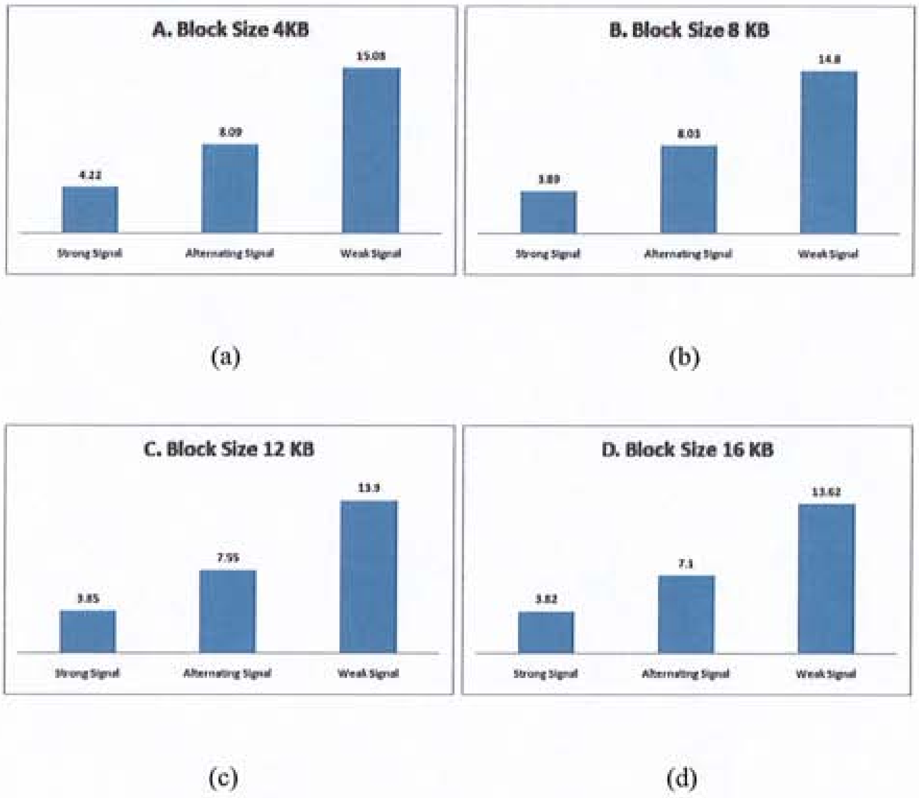


Figure 4.13: Adaptive scheme 25-75 binary file energy results

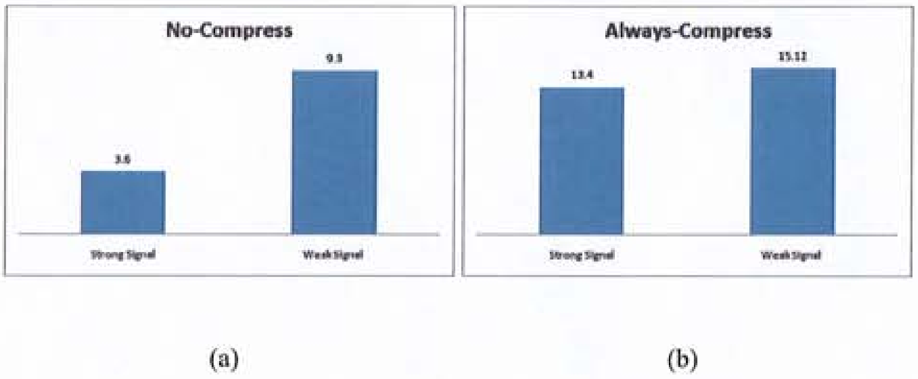


Figure 4.14: 25-75 binary file energy results

4.3.5 Results Evaluation

The energy results presented in the previous sections show that the performance of the always-compress and the no-compress schemes depends extensively on the compression ratio of the file to be sent. Both schemes exchanged outperformance; thus no scheme can be claimed better than the other. Furthermore, the outperformance of one scheme over the other, depending on the file, is immense and cannot be comparable. On the other hand, the adaptive scheme has shown remarkable gain in energy in most of the circumstances. This achievement can be traced to the adaptive nature of the scheme that combines the best out the always-compress and the no-compress (beats the former as it sends data compressed on the fly). It is true that with some files the other schemes scored slightly better than the adaptive scheme, but with only one setting (weak or strong signal). In general, the results demonstrate that the adaptive scheme is able to score effective energy gain without being significant affected by the compression ratio. Because a normal usage of a mobile device implies an alternation among the nature of the files to be sent, the adaptive scheme will be able to score energy gain that will impressively overcome all other approaches.

Moreover, a careful analysis of the collected results reveals that the compression ratio seems to have a notable influence on energy consumption. Fig. 4.15 shows the energy consumption results of the adaptive compression scheme (block size 4 KB) with the various binary files in correspondance to the compression ratio.

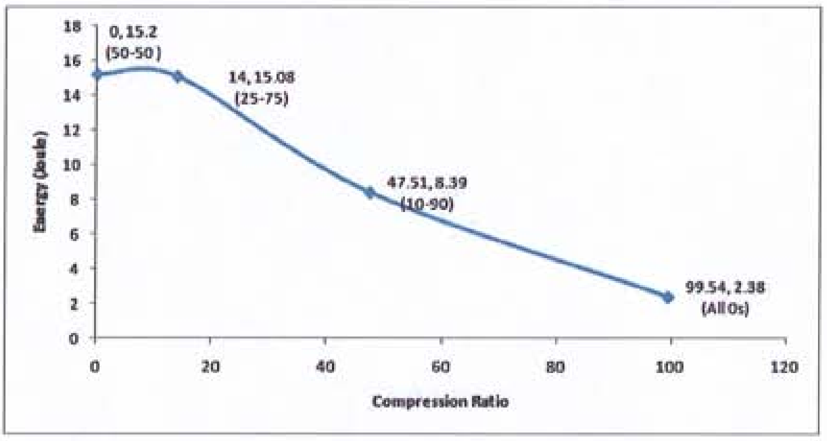


Figure 4.15: Energy consumption vs. compression ratio

It can be deduced from the plot in Fig. 4.15 that the energy consumption is closely related to the compression ratio. With the increase in the compression ratio, a decrease in energy consumption is noticed. Thus, the relation between the compression ratio and the energy consumption is inversely proportional.

4.4 Impact of Decompression on Energy Consumption

The energy results presented have shown that compression leads to energy saving for the transmitting node. However, the compressed data should be decompressed at the receiving node. Accordingly, it should be verified if the decompression process is saving energy as well for the receiving node or it is imposing additional energy requirements than simply receiving data uncompressed. For this purpose, an experiment was conducted. The 2.52 MB text file was sent between two PDAs while recording the energy of receiving data for the destination node in two scenarios at a signal strength level of -85 dBm. The first scenario consisted of sending the file uncompressed, thus the energy of receiving 2.52 MB of data was recorded. The second scenario consisted of sending the file compressed, thus the energy of receiving and decompressing 984 KB was recorded. The energy results for the two scenarios are shown in Fig. 4.16.

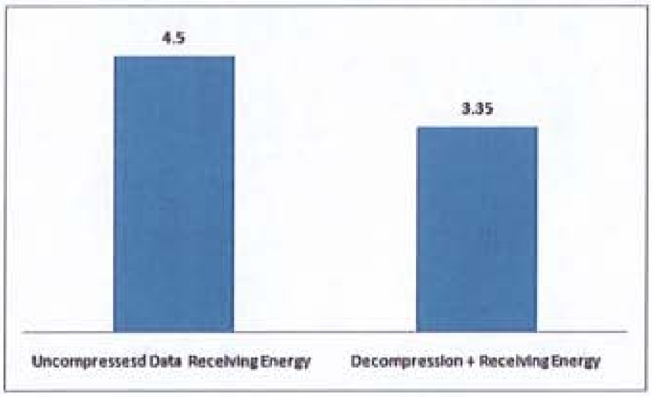


Figure 4.16: Decompression case energy results

The energy results show that receiving and decompressing the data has saved 25.55% of energy in comparison to receiving data uncopressed. Thus, it can be concluded that applying compression appropriately at the sending node as indicated by the adaptive scheme leads to energy gain at the receiving node through applying decompression.

Chapter 5

Energy-Aware Adaptive Compression over Multiple-hops

After verifying the profitability of the adaptive compression scheme between two nodes (source and destination), the performance in multiple-hops environment is evaluated. A multiple-hops environment is composed of several mobile devices that form a network. Each node can play the role of a sender that transmits data, a receiver that acquires data and a relay that forwards data. Actually, the communicating devices could form among them a Peer-to-Peer network to share files and information. Such a scheme of file sharing is attractive and expected to prevail due to the large availability of mobile devices among people. Thus, the adaptive scheme is evaluated in such an environment to investigate whether it leads to energy gains.

To evaluate the performance, a multiple-hops environment was simulated in which the adaptive compression, the no-compress and the always-compress schemes are applicable on data blocks that travel from a source, through intermediate forwarding nodes, to a destination. It should be noted that whenever a block of data is compressed at a given node, then the adaptive approach simply forwards the data at the proceeding nodes without any compression. To measure the sending energy within the multiple-hops environment, a transmission and compression empirical models to estimate energy consumption were derived from empirical data. Then, pilot runs were carried while recording the energy consumption for the three compression schemes.

5.1 Empirical Models

To estimate the energy consumption over multiple-hops, two empirical models were derived from empirical data. The first model serves as an energy estimate for the data to be sent. To collect the empirical data needed to form this model, blocks of data were sent between two PDAs while recording the per block energy and the signal strength level at which data were sent. For each signal strength value, the energy required to send a single byte was determined by dividing the enrgy of sending the block to the number of bytes per block as shown in Table 5.1.

Table 5.1: Energy required per byte at a given signal

Signal	Energy
-20	0.0001
-51	0.001007
-56	0.001027
-63	0.000929
-68	0.001154
-80	0.002205
-88	0.004604
-89	0.004633
-90	0.005038
-92	0.005143

The collected data was then fitted into an appropriate mathematical model. It is to be noticed that the energy required to send a single byte is increasing with the weakness of the signal. Furthermore, the increase in energy is not constant i.e. the energy increase between the interval [-40, -50] is more than the increase between the interval [-30, -40] implying a convexity property. Being increasing and convex, the best fit model for the above data is exponential. As shown in Fig. 5.1, $y = 0.00004\ e^{-0.052x}$ is the exponential model where the input *x* represents the signal strength value and output *y* represents the energy required to send a single byte at the inputted signal strength.

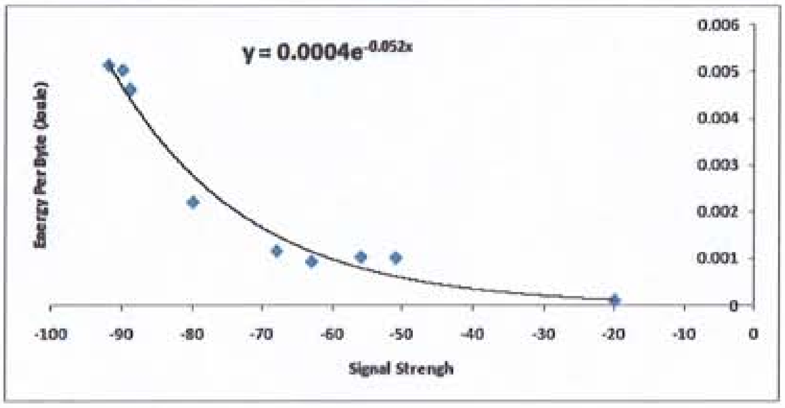


Figure 5.1: Empirical model to estimate the energy of sending data

To validate the transmission empirical model, a 2.5 MB file was sent at the signal strength of value -70 dBm. The real energy consumption for transmission was 3.10 Joule. Using the empirical model, the transmission energy is 3.96 Joule. Thus, the error rate for the transmission model is 13.92%.

While the first empirical model estimates the tranmission energy for a byte at a give signal strength value, a second analytical model was determined to estimate the energy required to compress data. To collect empirical data for this model, the 5.35 MB text file was compressed. The energy required to compress each block of data was recorded. Energy measurements were grouped as shown in Table 5.2. Each row consists of the cumulative size of the previous blocks in addition the current row block and the cumulative energy of compressing the previous blocks in addition to the current block.

The collected data turned out to fit a linear mathematical model. As shown in Fig. 5.2, $y = 0.0008x$ is the linear model where *x* represents the size of the data to be compressed in bytes and the output *y* represents the energy required to compress the data in Joule.

Table 5.2: Compression energy per blocks of data

Data Block (bytes)	Energy
0	0
533000	0.393
1066000	0.818
1599000	1.211
2132000	1.606
2665000	2.022
3198000	2.404
3731000	2.809
4264000	3.248
4797000	3.664
5330000	4.058

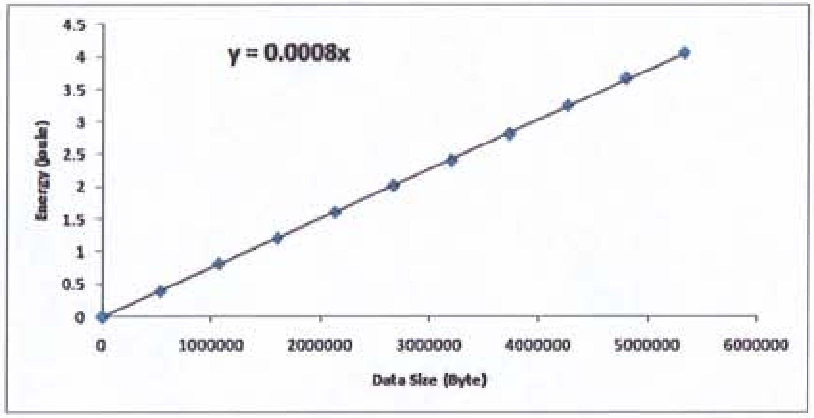


Figure 5.2: Compression empirical model

5.2 Simulation Setup

To assess the fruitfulness of the adaptive approach over multiple-hops, a simulation environment was initiated as depicted in Fig. 5.3. The environment consisted of 4 nodes connected together through a TCP connection with a fixed path, where the first node is sending 5.35 MB text file (same file used to determine the compression empirical model) to the fourth node. The energy of forwarding data to node 4 from nodes 1, 2 and 3 were recorded. The simulation environment encompassed the adaptive compression model, the no-compress model and the always-compress model. As $Total\ Energy = Transmission\ Energy + Compression\ Energy$, the empirical models were used to estimate the required energy of sending and compressing the data.

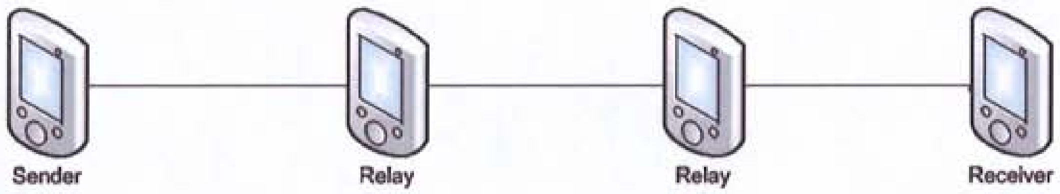


Figure 5.3: Multiple-hops environment

5.3 Simulation Results

The simulation process was run in five different scenarios. Each scenario exhibited different signal strength settings to simulate the various schemes in various conditions. For a strong signal, -54 dBm was set. For weak signal, -92 dBm was set. For an alternating signal, the levels were determined randomly. The simulation energy results for each scenario are presented below.

5.3.1 First Scenario: Weak Signal

The first scenario consisted of simulating a constant weak signal along the path from node 1 to node 4 as shown in Fig. 5.4. The energy required to compress and send the text file for the adaptive approach and the no-compress approach were recorded. The always-compress scheme was dropped in this scenario because the adaptive approach acts closely with a weak signal.

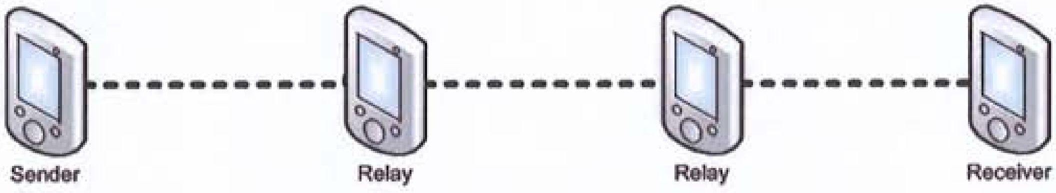


Figure 5.4: Weak signal strength along the path

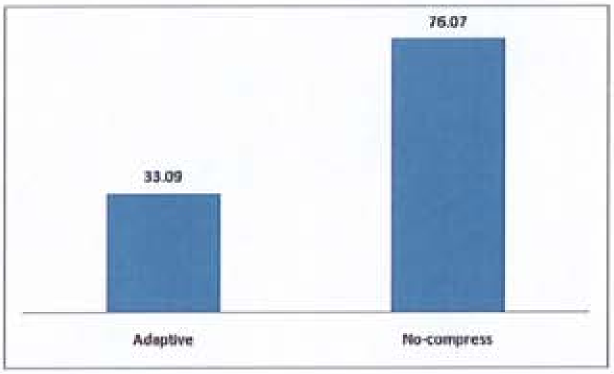


Figure 5.5: Adaptive vs. No-Compress

The first scenario energy results are shown in Fig. 5.5. It can be seen that the adaptive approach beats the no-compress approach by an energy gain amounting up to 129.85%. It is clear that the adaptive approach significantly outperforms the no-compress approach. The adaptive approach applies compression at the first node, making nodes 2 and 3 forward a small amount a data. However, the same amount of data at each node is forwarded with the no-compress approach.

5.3.2 Second Scenario: Strong Signal

The second scenario consisted of simulating a constant strong signal along the path from node 1 to node 4 as shown in Fig. 5.6. The energy required to send the text file for the adaptive approach and the always-compress approach was recorded. This time, the no-compress scheme was dropped because the adaptive approach acts closely with a strong signal.

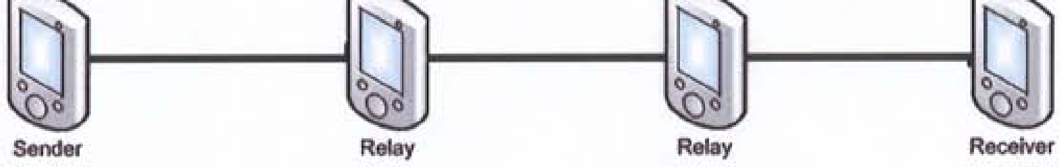


Figure 5.6: Strong signal strength along the path

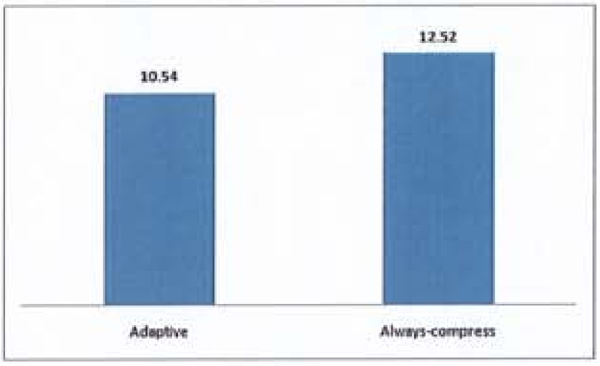


Figure 5.7: Adaptive vs. Always-Compress

The second scenario energy results are shown in Fig. 5.7. An energy gain of 16.23% were achieved by the adaptive approach over the always-compress approach. The relatively small amounts of energy gain in comparison to scenario 1 can be traced back to the fact that the always-compress approach compresses the file completely at node 1. As a matter of fact, node 2 and 3 does nothing but transmitting the small amount of compressed data. However for the adaptive approach, nodes 2 and 3 transmit the entire text file without applying any compression as the signal is strong. It can be concluded that the always-compress can be close to the adaptive approach with such a scenario, but with putting a penalty on the first node in the path as it is paying the entire amount of the compression energy.

5.3.3 Third Scenario: Random Signal

The third scenario consisted of simulating a totally random signal along the path as shown in Fig. 5.8 and of applying the three approaches together. Each block of data was sent by each approach at the same randomly generated signal strength. The aim from this setting was to make the signal strength fluctuate between a strong signal and a weak one.

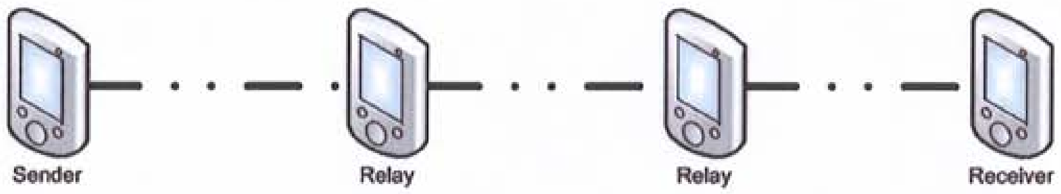


Figure 5.8: Alternating signal strength along the path

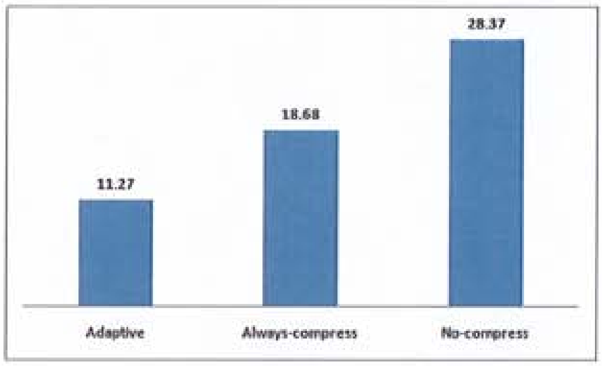


Figure 5.9: Adaptive vs. Always-Compress vs. No-Compress

The third scenario energy results are shown in Fig. 5.9. The adaptive approach beats the always-compress approach and the no-compress approach by 40.24% and 60.27% of energy gain respectively. Such results show the outperformance of the adaptive approach over other approaches. Once again the always-compress approach is the closer to the adaptive approach, but the burden of compression is putted on the first node. Fig. 5.10 shows the energy distribution among the nodes for current scenario. The first node has paid the most of the energy for the always-compress approach. The energy results of the adaptive approach show that the energy is better distributed among the participating nodes. Based on that, the adaptive approach is fair in distributing the load of energy consumption over the participating node. It could be the case that the adaptive approach would act as the always-compress approach if the signal between the first and second node is always weak, but it would still score better than the always-compress approach as it sends data compressed on the fly. It should be noted that the results of the third scenario are very significant because they are the closer to reality as the signal is fluctuating from weak to strong then to weak or remaining strong etc.

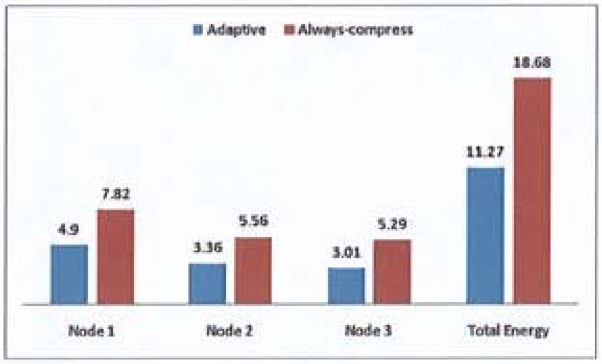


Figure 5.10: Energy load per node

5.3.4 Fourth Scenario: Alterating Signal

The fourth scenario consisted of simulating a signal that alternates between every two nodes. A fifth node was added to the simulation environment to achieve a weak-strong-weak-strong sequence of signal strength levels as shown in Fig. 5.11. The rational behind initiating this setting was to test the three compression schemes with a signal level that alternates infrequently.

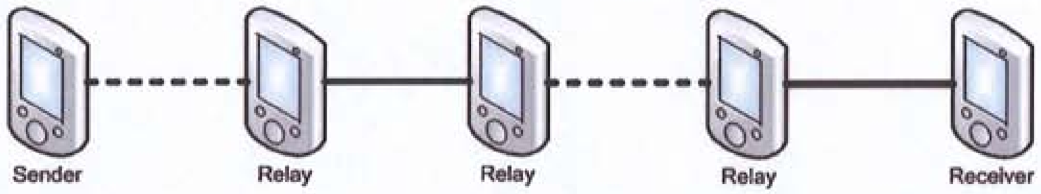


Figure 5.11: Weak-Strong-Weak-Strong signals strength along the path

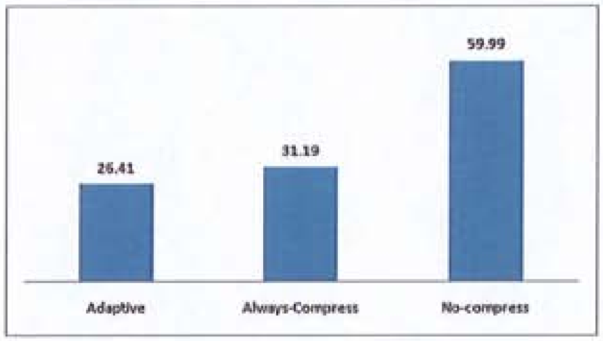


Figure 5.12: Adaptive vs. always-compress vs. no-compress energy results

The energy results illustrated in Fig. 5.12 show that the adaptive scheme beats the no-compress and the always-compress scheme by an energy gain amounting to

55.97% and 15.32% respectively. It is to be noted that the adaptive and the always-compress approaches acts similarly with the initiated environment's setting. However, the on-the-fly compression characteristic of the adaptive scheme makes it score higher energy gain. Moreover, the energy results for the different scenarios denote that the no-compress approach is not suitable for multi-hops environment. The adaptive and the always-compress schemes have beaten the no-compress scheme intensively. The only scenario that can make the no-compress scheme energy profitable is with a strong signal along the path.

5.3.5 Fifth Scenario: Testing with a 10-Hop Path

The fifth scenario consisted of investigating whether the number of nodes affects the energy results. It could be the case that with the increase of the number of nodes, the adaptive scheme might loose its superiority as the data to be transfered could reach a point where no compression could be applied. Such a scenario would make the adaptive approach acts similarly to the no-compress scheme after a certain number of hops. To evaluate the performance, the number of nodes in the multiple-hops environment was increased to ten while simulating a random signal along the path. the energy results are shown in Fig. 5.13.

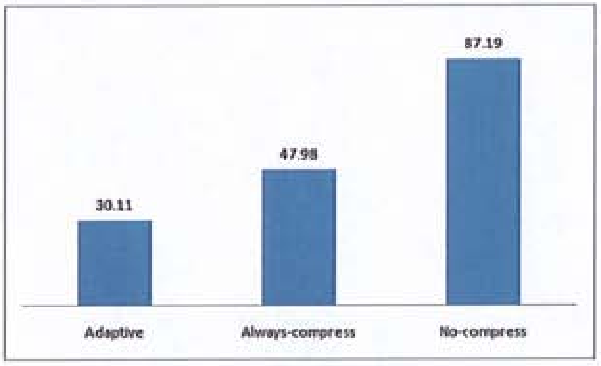


Figure 5.13: Energy results of the various schemes with 10 nodes

The energy results show that the adaptive scheme still beats the no-compress and the always-compress schemes by 65.46% and 37.24% of energy gain respectively. Though the increase of the number of nodes, the adaptive scheme leads to significant energy gain. As a matter of fact. The energy gains that the adaptive scheme acquire before reaching the point of simply forwarding compressed data is large enough to keep it ahead of the two other schemes. It is true that the adaptive scheme would reach the point of acting like the no-compress scheme after a certain number of hops, but with difference in the size of forwarded data. The no-compress scheme always forwards the original size of the data file while the adaptive scheme would forward the size of the compressed data. When it comes to the always-compress approach, it constantly applies compression then sends data irrespective of the number of nodes.

5.3.6 Results Interpretation

Once again, the adaptive scheme proved its superiority to the other two scheme. The adaptive approach seems to be a fit for multi-hops environments. Though the no-compress approach scores close to the adaptive, the first transmitting node is paying the total amount of the compression energy. The adaptive scheme is fair in distributing the load of energy consumption among the participating nodes making the batteries of the participating mobile nodes drain at a comparable rate. Furthermore, the no-compress approach has a weak performance when compared to the adaptive and always-compress scheme except with strong signal along the whole path.

Chapter 6

Conclusions and Future Work

The main contribution of this thesis is the proposition of an adaptive compression scheme for mobile devices. Energy measurements have shown that compression cannot be applied blindly. The computational requirements of compression consume more energy than sending data uncompressed with strong signal strength. The latter implies a high transmission rate with limited number of errors and retransmissions. On the other hand, the degradation of the channel quality with weak signal strength necessitates energy requirements that overcome the energy cost of compression. This makes compression an energy-profitable technique along weak signal strength. The adaptive compression scheme accounts for these observations and apply data compression only when energy gain is promising. For each block of data to be sent, the adaptive scheme checks for the signal strength and applies compression only when the signal is weak.

The profitability of the adaptive compression scheme was shown by means of experimental work. The adaptive scheme was tested against the always-compress and the no-compress schemes. The former applies compression a-priori to sending while the latter discard the compression option. In general, the energy results collected from the experimental work showed that the adaptive scheme takes the best out of the other two schemes. With a strong signal, the adaptive scheme acts similarly to the no-compress approach which makes it overcome the computational energy requirements of compression. With a weak signal, the adaptive scheme acts closely to the always-compress scheme and alleviates the channel degradation problems by sending less data. In addition, the adaptive scheme even beats the always-compress scheme with a weak signal. Though both schemes apply compression, the adaptive scheme narrow down significantly the idle time of the wireless card by applying compression on the fly. On the contrary, the always-compress scheme introduces a significant period of idle time because it requires data to be fully compressed ahead of sending it. With an alternating signal, the adaptive scheme sends blocks of data that alternates between being compressed and uncompressed, which results in important energy gains. Furthermore, the adaptive scheme showed its superiority in multi-hops environments. The no-compress scheme was way behind the adaptive scheme while the always-compress was much closer. However, the always-compress scheme always panelizes the first transmitting node with the burden of the entire compression energy. The adaptive scheme fairly distributes the compression and transmission energy over the participating nodes, which makes the battery drain rate of all nodes comparable.

As for future work, the adaptive scheme should be implemented as a standalone application. The implementation should incorporate the utilization of a dynamic block size. The block size can be increased gradually with the steadiness of the signal strength. When the signal weakens, the block size should be decreased. In addition, the scheme should automatically skip over compression for uncompressible data. In this realm, it is worth to investigate techniques that foretell the compression ability of a certain amount of data. The results of the various binary files encourage such an investigation. However, special care should be put to make these techniques to consume light energy. Furthermore, a work should be done to optimize compression algorithms is such a way that reduces energy consumptions and increases the compression ratio. The optimization should not be device centric, but should be universal and can fit all brand devices. Moreover, it was showed that decompression leads to energy gains. However, additional work should carried to verify decompression process.

Most importantly, the adaptive scheme application should be implemented to be an operating system service. The latter works in the background to monitor data transmissions and applies the adaptive scheme. The service should be granted the right to interact with the network transport layer to add information that flags compression and the compression algorithm type. Such information is needed for the intermediate nodes in order to forward data without compression and for the receiver node to decompress data with the appropriate algorithm.

Bibliography

[1] Zhang Y., Liu W., Lou W., and Fang Y. (2006), Location-based compromise tolerant security mechanisms in wireless sensor networks, *IEEE Journal on Select Areas in Communications, 24, 247-260.*

[2] Pering T., Agarwal Y., Gupta R., and Want R. (2006), CoolSpots: Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces, *Proceedings of the 4th international conference on Mobile systems, applications and services.*

[3] Li K., Nanya T., and Qu W. (2007), Energy Efficient Methods and Techniques for Mobile Computing, *Third International Conference on Semantics, Knowledge and Grid.*

[4] Sivalingam K. M., Chen J.C., Agrawal P., and Srivastava M. (2000), Design and Analysis of Low-power Access Protocols for Wireless and Mobile ATM Networks, *Wireless Networks, 6, 73-87.*

[5] Singh S., and Raghavendra C. S. (1998), PAMAS: PowerAware Multiaccess Protocol with Signalling for Ad Hoc Networks, *Computer Communication Review, 28, 5-26.*

[6] Minn H., Zeng M., Annamalai A. J., and Bhargava V. K. (2001), An Efficient ARQ Protocol for Adaptive Error Control over Time-Varying Channels, *Wireless Personal Communications: An International Journal, 17, 3-20.*

[7] Perkins C. E., Royer E. M., and Das S. R. (2000), Ad Hoc ondemand Distance Vector (AODV) Routing, *IETF Draft MANETWorking Group.*

[8] Johnson D. B., Maltz D.A., Hu Y. (2000), and J. G. Jetcheva, The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks, *IETF Draft MANET Working Group.*

[9] Bakre A., and Badrinath B. R. (1995), I-TCP: Indirect TCP forMobile Hosts, *Proc. of ICDCS (pp. 136-143).*

[10] Brown K., and Singh S. (1997), M-TCP: TCP for Mobile Cellular Networks, *Computer Communication Review, 27, 19-43.*

[11] Intel Corporation (2000), *Intel Power Measurement Tools.*

[12] Intel Corporation (2000), *Advanced Configuration and PowerInterface.*

[13] Flinn J., and Satyanarayanan M. (1999), Energy-aware Adaptation for Mobile Applications, *Proc. of SOSP (pp.48-63).*

[14] Merkel A., and Bellosa F. (2006), TBalancing Power Consumption in Multiprocessor Systems, *Proc. of EuroSys (pp. 403-414).*

[15] Zeng H., Ellis C. S., Lebeck A. R,, and Vahdat A. (2002), ECOSystem: Managing Energy as a First Class Operating System Resource, *Proc. of ASPLOS (pp. 123-132).*

[16] Balakrishnan S., and Ramanan J. (2001), Power-aware Operating System using ACPI, *CS736 Project.*

[17] Barr K., and Asonovic K. (2006), Energy-aware lossless data compression, *ACM Transactions on Computer Systems, 24, 250-291.*

[18] Xu R., Li Z., Wang C., and Ni P. (2003), Impact of data compression on energy consumption of wireless-networked handheld devices, *IEEE International Conference on Distributed Computing Systems.*

[19] Krintz C., and Sucu S. (2006), Adaptive on-the-fly compression, *IEEE Transactions on Parallel and Distributed Systems, 17, 15-24.*

[20] (2007) HP iPAQ hx2400 pocket PC series overview, Web Site: http://www.hp.com.

[21] (2009) National Instrument USB-6251, Web Site: http://sine.ni.com/nips/cds/view/p/lang/en/nid/202597.

[22] Lee R., and Nathuii R. (2000), Power and performance analysis of PDA architectures, *Advanced VLSI Computer Architecture.*

[23] (2007) Sharpziplib, Web Site: http://icsharpcode.net/OpenSource/SharpZipLib.

[24] Chandra S. (2003), Wireless Network Interface Energy Consumption Implications of Popular Streaming Formats, *Multimedia Systems, 9, 185-201.*

[25] Narendran B., Sienicki J., Yajnik S., and Agrawal P. (1997), Evaluation of an Adaptive Power and Error Control Algorithm for Wireless Systems, *Proc. of ICC (pp. 349-355).*

[26] (2009) Bzip2, Web Site: http://www.bzip.org/.

[27] (2009) Info-ZIP Home Page, Web Site: http://www.info-zip.org/.

[28] Huffman D.A. (1952), A Method for the Construction of Minimum-Redundancy Codes, *Proceedings of the I.R.E. (pp. 1098-1102).*

[29] Sohraby K., Minoli D., Znati T. (2004), Ad Hoc Wireless Networks: Architectures and Protocols, *Prentice Hall Communications Engineering and Emerging Technologies Series.*

[30] Ziv J., and Lempel A. (1978), Compression of individual sequences via variable-rate coding, *IEEE Transactions on Information Theory, 530- 536.*

[31] (2009) GZIP homepage, Web Site:
http://www.gzip.org/.

[32] (2009) Deflate, Web Site:
http://opensource.franz.com/deflate/index.html.

[33] (2009) BitTorrent, Web Site:
http://www.bittorrent.com/.

[34] (2009) LimeWire, Web Site:
http://www.LimeWire.com/.

[35] (2009) Kazaa, Web Site:
wwww.kazaa.com/.