HELSINKI UNIVERSITY OF TECHNOLOGY
Faculty of Electronics, Communications, and Automation
Department of Communications and Networking

Le Wang

Evaluation of Compression for Energy-aware Communication in Wireless Networks

Master's Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

Espoo, May 11, 2009

Supervisor: Professor Jukka Manner

Instructor: Sebastian Siikavirta

HELSINKI UNIVERSITY OF TECHNOLOGY ABSTRACT OF MASTER'S THESIS

Author: Le Wang	
Title: Evaluation of Compression for Energy-aware Communication in Wireless Networks	
Number of pages: 75 p.	**Date:** 11th May 2009
Faculty: Faculty of Electronics, Communications, and Automation **Department:** Department of Communications and Networks	
Code: S-38 **Supervisor:** Professor Jukka Manner **Instructor:** Sebastian Siikavirta	

Abstract

In accordance with the development of ICT-based communication, energy efficient communication in wireless networks is being required for reducing energy consumption, cutting down greenhouse emissions and improving business competitiveness. Due to significant energy consumption of transmitting data over wireless networks, data compression techniques can be used to trade the overhead of compression/decompression for less communication energy.

Careless and blind compression in wireless networks not only causes an expansion of file sizes, but also wastes energy. This study aims to investigate the usages of data compression to reduce the energy consumption in a hand-held device. By conducting experiments as the methodologies, the impacts of transmission on energy consumption are explored on wireless interfaces. Then, 9 lossless compression algorithms are examined on popular Internet traffic in the view of compression ratio, speed and consumed energy. Additionally, energy consumption of uplink, downlink and overall system is investigated to achieve a comprehensive understanding of compression in wireless networks. Moreover, we also discuss the relation between file contents and wireless network status in the perspective of energy consumption and propose proper ways to deploy compression in energy-aware communication.

Keywords: compression, power, energy, awareness, wireless, efficiency, ratio	**Language:** English

ACKNOWLEDGEMENT

My deepest gratitude goes first and foremost to Professor Jukka Manner, my supervisor, for his illuminating instructions and guidance. Without his constant support, this thesis never comes into a reality. Moreover, this work was a part of the Future Internet program of TIVIT (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT), thus, I also would like to take this chance to express my gratitude to the support from TEKES.

Second, I would like to express my sincere thanks to a number of people who ever offered their invaluable assistance in the preparation of this thesis. Special appreciation is extended to Nokia for providing the necessary hardware in this study. I also would like to thank Arto Karppanen, Yu Xiao and Eero Sillasto, whose help and suggestions paved a smooth way for this thesis to come into being. My appreciation also goes to my instructor, Sebastian Siikavirta and others who offered help.

Here, I own special thanks to my beloved girlfriend, Du, who always supports and encourages me. So thank you my Love for being in my life.

Finally, my thanks would go to my family for their unconditioned love and endless support over years. Especially, this work is dedicated to my mother for her 55th birthday and her endless love.

Le Wang

11.05.2009

LIST OF ACRONYMS

BMP	Bitmap
BWT	Burrows-Wheeler Transform
CCITT	the facsimile standard, Group 3 or 4
HTML	HyperText Markup Language
HSDPA	High-Speed Downlink Packet Access
ICT	Information and communication technologies
ITU	International Telecommunication Union
JPEG	Joint Photographic Experts Group
JTG	Jugi’s Traffic Generator
LZ	Lempel-Ziv
LZFG	Lempel–Ziv–Fiala-Daniel
LZMA	Lempel–Ziv–Markov chain Algorithm
LZMW	Lempel–Ziv-Miller-Wegman
LZO	Lempel–Ziv–Oberhumer
LZP	Lempel–Ziv-PPM
LZRW	Lempel–Ziv-Ross-Williams
LZS	Lempel–Ziv–Stac
LZSS	LZ-Storer-Szymanski
LZW	Lempel–Ziv–Welch
LZY	Lempel–Ziv-Yabba
MPEG	Moving Picture Experts Group
MP3	MPEG-1 Audio Layer 3
MVRCA	Magnalink Variable Resource Compression Algorithm
NTP	Network Time Protocol

PDF	Portable Document Format
PGF	Progressive Graphics File
PPM	Prediction by partial matching
PPP	Point-to-Point Protocol
QoS	Quality of Service
RLE	Run-Length Encoding
SHOK	Strategic Centres for Science, Technology and Innovation (Finnish acronym)
SWF	Shockwave Flash
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
WLAN	Wireless local area network
WMA	Windows Media Audio
XML	Extensible Markup Language

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 INTRODUCTION

The Internet, a global integration of physical infrastructure, software and information, is arguably one of the most important inventions in last decades. Internet is used today to spread and search for information, share knowledge, interact with each other, execute commercial operations, play games and so on. Nowadays the influences have been spread over the world. It has been said that no one's life is without being influenced by Internet no matter how he or she interacts with it, directly or indirectly.

Looking backward to the history, Internet was just born as connections of work stations in a network in the 1960s, but it quickly spread and penetrated into the whole world only within the following two decades and was covered with innumerable fortune and glory. However, Internet was also born with technical limitations which evolve side effects on social and economic functions. That is why redesign of the Internet architecture has been taking shape in across the globe.

Wireless networks have had a significant impact on the world and are, by any measure, the fastest growing segment of telecommunications. Already today, there are almost 1.5 billion Internet users and this number may go up to 4 billion as the Internet becomes genuinely mobile (ITU-T. 2009-09-25). Therefore, the main interest of this thesis closely ties to mobility.

Particularly, in the aspects of energy efficiency and energy awareness of wireless networks, ICT-based energy efficiency is considered on one hand for substantial savings in energy production, reducing greenhouse gas emissions, enhancing business competitiveness and social welfare. So far, Internet traffic grows by a factor of roughly 10 every 5 years which follows Moore's Law and the price paid for the growth is a doubling of energy consumption every 5 years (Nemertes Research). According to Efore Oy, the energy consumption of ICT infrastructure is 2.1 TWh (billion kWh) and energy consumption of ICT user terminals is 4.6 TWh in Finland. Furthermore, the corresponding greenhouse gas emission contributed by ICT is approximate 2.5% of carbon dioxide and other greenhouse gas emissions from all electricity generation technologies. Especially mobile user energy consumption is approximate 29kWh annually and the equivalent carbon dioxide is 55 kg (Reijo Mäihäniemi. 2008). The aforementioned numbers stress more emphasis on energy efficiency because of the needs to cut down costs, and additional reward is to lower greenhouse gas emissions for a climate improvement and take credit of positive image creation. In particular, the future Internet will not only offer energy efficient ICT devices and facilities but also provide ICT solutions for better energy savings.

On the other hand, stress is also laid on the importance of energy awareness. In a wireless network, hand-held devices are being crammed with a large number of communication capabilities like UMTS, HSDPA, IEEE802.11b/g and Bluetooth, and an ever-increasing range of communication services, which result in an important design factor for manufactures: a dramatic and critical increase of energy consumption in the devices. Energy consumption directly affects functionality and usability of the devices. Given the initial energy level for every device, Quality of Service (QoS) and lifetime are typically considered as two main criteria to evaluate network services, unfortunately the battery capacity is a fundamental limitation, thus the motivation to emphasize energy awareness is the optimization of system's design at all layers.

In order to create solutions for energy efficiency and energy awareness of the future Internet communication, novel techniques are needed. Based on a study of Kenneth Barr and Krste Asanovic (Barr. K. C. & Asanovic. K 2006), the energy consumed on a single bit transmission over wireless is over 1000 times greater than a single 32-bit CPU computation. This fact shows great benefits in reducing the number of bits transmitted over the radio link and offers potentials for reducing transmission time and energy consumption in mobile and wireless access networks. In their paper, bzip2, compress, lzo, ppmd and zlib are examined with experimental work, which indicates that energy savings from compression are not only depended on compression ratio but compression speed due to the tradeoff between the energy consumption of intensive computational requirements for reducing file sizes, and the energy saved by less bits to be transmitted. Besides, the paper proposes an asymmetric compression using different compression and decompression schemes to save more energy.

In another study (R Xu, Z Li, C Wang, and P Ni. 2003), the researchers focus on saving energy on mobile devices for only downloading. It requires the server to compress data with certain schemes to achieve the maximum energy saving in mobile devices. Their proposal to reduce energy is to decompress data on-the-fly instead of decompressing entire data after receiving them all.

Moreover, there is a study (Maddah Rakan, Sharafeddine Sanaa. August 2008) taking the signal strength into consideration. The approach in the study is to compress data on-the-fly only when the signal strength is good enough. This point is also mentioned in this paper (C Krintz and S Sucu. January 2006) that compression used or not is based on the network status and bandwidth availability. Regarding the transmission impacts, there is a study (J.-P. Ebert, B. Burns, and A. Wolisz. February 2002) investigating impacts on energy consumption of IEEE 802.11 network interfaces from the transmission rate, the RF transmission power and the packet size.

Overall, compression decreases the number of transmitted bits resulting in reduction of the transmission time. This introduces reduction in energy consumption as well. However, compression schemes involve tradeoff due to the intensive computation and memory access to compress and decompress data. The consequence might be that more energy is consumed than when simply transmitting the raw data. Furthermore, transmission rate in wireless networks may give different results in energy consumption and affect the decision on whether to deploy compression schemes or not.

Energy consumption on hand-held devices differs from each other due to hardware and software related factors, therefore an evaluation over modern hand-held devices provides a more timely understanding of data transmission and compression in the view of energy efficiency, and it is also possible to offer a chance to explore new approaches for more energy savings. Nokia N810 supports IEEE802.11b/g, thus a study of impact of transmission bit rate on energy consumption is carried on WLAN networks in this thesis. This study investigates the use of data compression, and evaluates its impact on energy consumption. Compression gives benefits in reduction of file size most of the time, however file types result in different compression effects as well as different energy consumption. Therefore, a large number of files with different types are examined to avoid careless misusage of compression programs or incorrectly choosing the compression programs on certain data. We also deeply examine 9 tools which are the representations of lossless compression schemes in several families of compression algorithms. The criteria we used to evaluate compression schemes are compression/decompression time and energy, and compression ratio. In the whole thesis, the term compression ratio is defined as the ratio between the uncompressed size and the compressed size. This study shows by means of experimental work that content-aware compression schemes reduce energy consumption significantly, while blind or careless use of compression results in a huge energy loss. Moreover, link status is an important factor to guide a proper compression usage.

The rest of this thesis is organized as follows. In the following chapter, compression algorithms are reviewed and the compression applications we evaluated are presented. Chapter 3 describes test equipment, experimental setup and workload, and explains the methodology of this study. The experimental study starts from Chapter 4. Chapter 4 investigates transmission bit rate impact on energy consumption. In Chapter 5, we examine compression ratio, compressed time and consumed energy of different compression software on a wide-range of workloads under different bit rates, and try to indicate which tools are capable of energy-aware lossless compression schemes. Besides, we demonstrate the benefits of compression gained when surfing Internet. In the last chapter, Chapter 6, we conclude our findings and suggest topics for future study.

Chapter 2 COMPRESSION ALGORITHMS

Compression is a technique to reduce the consumption of expensive resources. Normally, it helps to release the pain of quickly increased mass storage, lower the amount of data transmitted and reduce file size for improved program speed, therefore the criteria to evaluate compression algorithms are based on their compression speed, compression ratio and computational complexity. However, the research purpose of this study is to reduce energy consumption and achieve energy-aware communication in mobile devices, thus the solutions for our goal can not be concluded with reliance on the normal criteria and it is necessary to review state-of-art compression algorithms and compression programs to understand the behavior of compression for the purpose of energy savings.

Data compression algorithms can be divided into two categories, namely, lossy and lossless compression. Since lossy compression suffers from introduced differences when compressing and decompressing, a review of lossy compression is briefly introduced and we focus mainly on lossless compression algorithms, which allows compressed data to be reconstructed exactly identical to the original data. In order to present comprehensive evaluation of compression schemes, this study hence examines a wide range of compression algorithms that have been proposed for coding with an emphasis on the following compression algorithms: statistical compression, dictionary-based compression, predictive coding and context mixing.

Additionally, this chapter introduces the compression programs we used in the study, which are chosen from the most representative in each category mentioned above. In order to find a good solution in energy-aware communication, we take more compression programs into consideration to study their influence on energy consumption in our experiments.

2.1 Compression categories

Compression is a process to generate a representation of information using encoding schemes that require additional bits to recover the original information. Based on the ways to reconstruct encoded information, compression algorithms are categorized into two classes, as mentioned, lossless compression algorithm and lossy compression algorithm.

Lossless compression is used to represent information which can be recovered into the original data without any mismatch. Lossless compression thus provides ability to compress data which have to remain the same as the original data. Normally, text compression and compression used in communication are the cases that can not tolerate any difference between the original and reconstructed data.

Lossless compression takes use of reducing statistical redundancy in data, however this property results in a limitation that lossless compression fails to compress already compressed data and random data without any discernible patterns. Even more, misusage of lossless compression therefore gives an expansion to data size.

The term lossy compression is in contrast to lossless data compression, which can not recover or reconstruct the compressed data without difference, in exchange for better compression ratio. Lossy compression involves some loss of information, but in certain degree the lack of exact reconstruction can be tolerated. Actually, limited discernment and sensitivity of humans are made use of by lossy compression, thus the size of reconstructed data would be highly reduced if only the quality of recovered data does not bring in uncomfortable infidelity. More generally, lossy compression is widely used to compress voice and multimedia data. For example, WMA and MP3 are used to compress music, while JPEG and PGF are for image compression, and MPEG, H.261, H.263 and H.264 are designed for video (AMI Power Limited).

We are aiming to achieve energy efficient communication in wireless networks, which demands the transmitted data to be identical to the original, therefore the compression algorithms reviewed in this study are lossless compression. However, we mainly focus on the coding schemes used in our experiments. The theory study in this thesis is partly based on the books "Data compression: the complete reference" (David Salomon, Giovanni Motta, David Bryant. 2007) and "Introduction to data compression" (Khalid Sayood. 2000). The details of the algorithms are presented in the following sections.

2.2 Statistical Compression

Statistical compression schemes substitute symbols represented by codes to match code lengths with the probabilities of the symbols. In more details, statistical compression replaces the symbols which have a higher probability of occurrence by assigning variable-length codes to the symbols. Shorter presentations are used in the data resulting in reduced number of bits per symbol. Huffman coding and arithmetic coding are typical statistical compression schemes, which are introduced in the following sub-sections.

2.2.1 Huffman Coding

The Huffman method (D.A. Huffman. 1952) was developed by David Huffman. As a kind of statistical compression algorithm, Huffman coding replaces symbols in a file by a variable-length code table as well, but the difference is Huffman coding uses a particular way to derive a prefix code that expresses the most common characters using shorter strings of bits than are used for less common source symbols. Huffman coding, also a procedure for optimum prefix codes, follows the principles that symbols having a higher probability of occurrence will have shorter representation than symbols having a lower probability of occurrence in order to shorten the average number of bits per symbol.

In detail, the method starts to construct a tree of all symbols in descending order of their occurrence probabilities. Then the procedure carried on in following steps. The two symbols from the bottom of the tree are selected, and replaced with an auxiliary representation for both of them. Then the probabilities of the symbols are added to the top of the partial tree. The representations of each symbol are determined steps continue till the top of the tree.

Huffman coding is commonly used for data compression and also the basis for many other compression schemes. However, Huffman coding has some disadvantages, such as changes in optimal coding as the ensemble changes, and running through the entire file in advance. In order to address the problems, the improvements are brought in by some of the schemes. According to "Code and Parse Trees for Lossless Source Encoding" (Abrahams, J. Jun 1997), variations of Huffman coding include:

Adaptive Huffman coding

The standard Huffman coding has to take all statistic information of encoded symbols into consideration. However, the statistic information is known in advance normally. A variation called adaptive Huffman coding allows to avoid transmitting statistics data by calculating the probabilities dynamically based on recent actual frequencies in the input stream.

Adaptive Huffman coding starts with an empty Huffman tree and to modify the tree as symbols are being read. As a result, the Huffman tree is constructed just based on the statistic information already processed. This method avoids repeat reading of the entire source stream, therefore the advantage of Adaptive Huffman Coding is the encoding can be done in real time but the cost is that the scheme is more sensitive to the transmission errors.

n-ary Huffman template algorithm

The weights are often used in Huffman coding to represent numeric probabilities. The n-ary Huffman template algorithm uses the {0, 1, ..., n-1} alphabet encode message and allows any kind of weights , such as costs and frequencies, including non-numerical weights.

Huffman coding with unequal letter costs

In the standard Huffman coding, a problem named prefix coding problem is to find a prefix free code over minimum weighted average codeword size. Huffman coding with unequal letter costs is to minimize the weighted average codeword length.

The canonical Huffman code

The advantage of a canonical Huffman tree is that one can encode the codebook in fewer bits than the tree in standard Huffman coding since it provides an efficient way to store the codebook.

2.2.2 Arithmetic Coding

Equivalently to the Huffman coding, arithmetic coding evaluates statistics information with certain symbols and generates ideal variable-length codes as well. As opposed to Huffman coding, arithmetic coding represents the entire data set by a single number with a range between 0 and 1. This range is divided into sub-intervals each representing a certain symbol. The principle of arithmetic coding to assign a symbol with a number in the sub-intervals based on its probability of occurrence. The size is proportional to the probability, so the higher probability will be assigned a higher range.

The idea in arithmetic coding is to map the source sequence $u_1 u_2 \ldots$ into a number x in the interval on the real line which can be represented by the following equation,

$$x = \sum_{n=1} x_n 2^{-n}$$

where x_n is 0 or 1 for each n. This mapping ensures the random variable x uniformly distribute on the real line and each x_n (n=1, 2….) is independent and equiprobably equal to 0 or 1 (Robert G.Gallager, 1994).

Arithmetic coding starts to read symbols from source stream, and then assigns an interval to each symbol according to its probability. A narrower interval is represented by more bits, so this is the way the scheme used to achieve compression. The way to assign the interval is that a high-probability symbol always receives a larger interval while a low-probability symbol results in a narrower interval.

2.2.3 Summary

In this sub-section, as the most commonly used and efficient coding schemes, Huffman coding and its variations are described first and we also present another statistical compression method, arithmetic coding, is similar to Huffman coding.

One of the differences is that arithmetic coding encodes an entire message into one signal number rather than Huffman coding representing each symbol into a series of number. Another difference is that Huffman coding only produces optimal variable-size codes when the symbols have probabilities of occurrence that are negative powers of 2. This is because the Huffman method assigns a code with an integral number of bits to each symbol in the alphabet. Moreover, a worst case for Huffman coding is when the probability of a symbol exceeds 0.5 due to the upper limit of inefficiency unbounded (David Salomon, Giovanni Motta, David Bryant. 2007).

Considering the demerits of Huffman coding, arithmetic coding is more efficient or at least identical to Huffman coding, but in return, it demands high computation and runs slowly and does not produce prefix code. Additionally, US patents cover a variety of techniques for arithmetic coding. So, arithmetic coding does not outstand over Huffman coding in practice.

2.3 Dictionary Compression

Unlike statistical compression, dictionary-based compression methods operate on searching matched strings and replacing the strings' positions by references instead of using variable-length codes to represent symbols in source stream based on their probabilities of occurrence. A typical example, text compression, shows that some words recur constantly in a text source and there are some other words do not or rarely occur, thus it is reasonable to encode the frequently occurred words with references, which are generated in a dictionary containing commonly occurring words. If the word can be found in the dictionary, it is replaced by a reference; if it can not be found, the word is kept the same.

The dictionary-based compression is most useful with sources that generate a relatively small number of patterns quite frequently, such as text, and also has a good performance for some binary data such as image and audio data. According to the ways to construct the data structure, dictionary-based compression can be divided into two classes, static dictionary and adaptive dictionary compressions.

2.3.1 Static Dictionary

Static-dictionary-based compression, as its name implies, involves static dictionary decided before performing compression. This compression method is always used and most appropriate when knowing the source in advance. It is highly efficient to encode a source stream based on a static dictionary containing the recurring patterns if some knowledge about the source is known ahead of time. Therefore, static dictionary-based coding is narrowed to well work on some specific data or applications. Expansion of the data may occur instead of compression if the coding method is used in other situations. It also makes sense that static-dictionary-based coding may, however, be a good choice for a special purpose but is not useful for general purposes.

2.3.2 Adaptive Dictionary

Compared to static-dictionary-based compression, adaptive-dictionary-based coding is more generic and preferable. The idea behind this method is to use already processed data as source to form a dictionary that evolves over time. For more details, it starts the process with a default or a size-limited dictionary, and then continually read words from a source stream. Meanwhile, it splits the stream into chunks where word is compared with the word in the dictionary. If a matched word is found, a reference to it is written to a compressed file. Otherwise, the word is added to the dictionary. Under this circumstance, an old word should be deleted from the dictionary in order to keep the dictionary small for efficiency.

Although adaptive dictionary-based coding is more complex than static dictionary-based one, it still surprises us with following advantages (David Salomon, Giovanni Motta, David Bryant. 2007).

(1) String search and match operations are involved, rather than numerical computations.

(2) The decoder can easily reconstruct dictionary and does not have to parse the input stream in a very complex way and search the dictionary to find matches.

The fundamental works on adaptive-dictionary-based coding started with two papers of Jacob Ziv and Abraham Lempel (Jacob Ziv, Abraham Lempel. 1977 and 1978), in which two approaches and their variations are provide to construct an adaptive dictionary. Named by the year, the approaches launched in 1977 were categorized to the LZ77 family, while the approaches mentioned in 1978 paper were classified to the LZ78 family correspondingly.

LZ77

Known as a “sliding window” compression algorithm, LZ77 constructs the dictionary based on the previously encoded sequence. The encoder maintains the window consisting of two parts, namely, a search buffer which is the current dictionary containing the recently encoded sequence, and a look-ahead buffer that contains the sequence to be encoded. The window is shifted through the source stream as the symbols are being encoded. The encoder scans the entire search buffer for a match to the first symbol in the look-ahead buffer. When consecutive symbols in the search buffer match the symbols in the look-ahead buffer, the length of the match is retrieved. The longest match selected by the encoder, along with the offset which is the distance of a search pointer for a match in the search and look-ahead buffer, and the codeword corresponding to the symbol in the look-ahead buffer that follows the match, consists a token, which is written to the output stream. In contrast, the decoder maintains a buffer with same size as the encoder's window. When the decoder decodes the input stream, it inputs a token to scan for a match in the buffer. Then it writes the match and the codeword to the output stream.

Due to improvements on the length of the match, buffer size, the offset and sliding window in the following years, many variations were announced, such as

LZSS(LZ-Storer-Szymanski), LZFG(Lempel–Ziv–Fiala-Daniel), LZW(Lempel–Ziv–Welch), LZMW(Lempel–Ziv-Miller-Wegman), LZY(Lempel–Ziv-Yabba), LZRW(Lempel–Ziv-Ross-Williams), LZS(Lempel–Ziv–Stac), LZO(Lempel–Ziv–Oberhumer), LZP(Lempel–Ziv-PPM), LZMA(Lempel–Ziv–Markov chain Algorithm).

Of special notice is the hash table employed by the Deflate algorithm to search for matches (P. Deutsch. May 1996).

Due to US patents on parts of LZ78 algorithm, LZW, a modification of LZ78, is described in the following sub-sections, as well as other well-known algorithms used by the compression programs in our experiments.

LZW

Lempel-Ziv-Welch (LZW) as an improved implementation of LZ78, was developed by Terry Welch in 1984. LZW concatenates input symbols one by one to a string which is searched for a match in the dictionary. As long as the match is found, the encoder adds a new symbol to the string till the match fails. The last successfully matched string is written to the output stream and the failed string is added to the next available dictionary entry. Furthermore, the design goal of LZW is not optimum but fast compression.

LZO

Lempel-Ziv-Oberhumer (LZO) is a popular variation of LZ77 originally published in 1996, and named by Markus Franz Xaver Johannes Oberhumer with Abraham Lempel, Jacob Ziv together. Like LZW, LZO is also designed for speed and suitable for real-time compression and decompression. LZO compresses and decompresses on a block of data with same size. It compresses the block into matches and runs of non-matching literals to produce good results on highly redundant data and deals acceptably with non-compressible data (Markus Franz Xaver Johannes Oberhumer. 2008).

LZP

Another variation of LZ77 developed by Charles Bloom is LZP, in which the P stands for “prediction”. LZP is based on LZ77 and the principle of context prediction. Actually, the recent encoded con-text is used as historical data to predicate the following data. LZP scans the most recent occurrence of the context and compares the context with the current input, which is encoded if a match is found, while non-matched context is written using another method.

LZMA

Lempel-Ziv-Markov chain algorithm (LZMA), having been under development since 1998, combines LZ77 and Deflate compression algorithm. It provides a high compression ratio and also high speed of decompression.

2.4 Predictive Compression

As mentioned above, dictionary-based compression methods encode symbols based on the dictionary. Statistical compression methods are based on the study on statistic information to construct suitable model and then actually encode the symbols according to their probabilities.

Furthermore, the models are built using two different approaches. One is based on assigning probabilities to the symbols, and this is also the approach used in statistical compression method. Predictive compression considers the context of a symbol when assigning it a probability. Actually, predictive compression methods examine the history of the symbols to get better knowledge about the symbols being encoded. Besides, statistical and dictionary-based compressions result in a more skewed set of probabilities in encoded messages. Predictive compression is one way to represent the message that would result in greater skew.

Indeed, already known context implies the knowledge about the following context to be encoded, which can be used to enhance compression performance with a cost of significant demand of memory. Three predictive compression methods, namely prediction with partial matching, Burrows-Wheeler transform and context mixing will be described in the following paragraphs.

2.4.1 Prediction with Partial Match

Prediction by partial matching (PPM), proposed by Cleary and Witten in 1984, is an adaptive statistical compression method based on context modeling and predicting the next symbol. The principle of PPM is searching the symbol to be encoded in a size reduced context for a match.

In more details, PPM starts with an order-N context, in which N indicates the context is the entail one. It searches its data structure for a previous occurrence of the current context followed by the next symbol. If the symbol has not previously been encountered in this context, an escape symbol contained in encoder's alphabet is used to encode, and the algorithm reduces the order from N to N-1. This process continues to find a match in the next smaller context, which is consisted of the rightmost $N - 1$ symbols of previous context until either a match is found, or a conclusion is drawn that no match has been encountered previously in any context. During the process, the encoder determines the probability that the symbol will appear following the particular context based on input data that has been seen in the past. The encoder then invokes an adaptive arithmetic coding algorithm to encode the symbol with the probability.

Although PPM compression is one of the best-ratio achieved lossless compression algorithms for text compression, it unfortunately requires a significant amount of memory for processing. Additionally, PPM does not work well for images since a digital image is normally the result of digitizing an analog image.

2.4.2 Burrows-Wheeler Transform

BWT, namely the Burrows-Wheeler Transform, transforms a block of data into well sorted format which keeps the same data elements as the original ones, only the orders is changed. The algorithm was invit-ed by, of course, Michael Burrows and David Wheeler in 1994, while the transform as a major part of this algorithm was developed by Wheeler in 1983.

Burrows-Wheeler Transform works on blocks unlike other compression algorithms and each block is encoded separately as a string. Thus, this compression method is also well-known as a block sorting com-pression. The idea behind the algorithm is summarized as follows. Given a sequence of length N, totally N sequences can be created by cyclically shifting one symbol to the left including the original se-quence itself. Then these N sequences are reordered lexicographically. By choosing the last letter of each sequence, a new sequence denot-ed by L consisting of letters is formed. Some more information denot-ed by I is also created to be used later by the decoder. The encoder compresses and writes L and I to the output stream starting with run-length encoding and then uses move-to-front coding, which is particu-larly effective on the type of structure exhibited by the sequence L, and applies Huffman coding at last. The decoder applies a reverse way in the encoder to decompress.

Burrows-Wheeler Transform is a general compression method. It achieves very high compression ratio on text, sound, and images, but it is extremely CPU and memory intensive in return.

2.4.3 Context Mixing

Context mixing, the last algorithm we examine, combines an arithmetic coder and a predictor which computes the next symbols by weighted averaging. As a predictive compression algorithm, it is related to Prediction by Partial Matching but the difference is that the next symbol is predicted by two or more combined statistical models to yield a more accurate prediction than any of the individual predictions.

Some compression methods are difficult to classify to the classes discussed so far as statistical, dictionary and predictive compressions. So, this section leaves a space for an unclassified compression method: symbol ranking. Symbol ranking is developed using techniques derived from LZ-77 and with a fast string-matcher. It orders a list of possible symbols from most likely to least likely based on current context and then encodes the position of the symbols in the ordered list (Peter Fenwick. 1996).

2.6 Compression programs

High compression ratio and fast compression/decompression speed are normally taken as merits to evaluate compression algorithms. Moreover, energy consumption of each algorithm is time related and also depends on memory usage. In order to find a good solution in energy-aware communication, we choose the most representatives in each category and also take compression programs' maturity and popularity into consideration. The compression programs we evaluate in our study are introduced as follows:

2.6.1 gzip

gzip (gzip homepage. 2009) is a well-known compression program based on deflation algorithm which is a combination of LZ77 and Huffman coding. It was first publicly released on October 31, 1992 and has been adopted by many applications.

gzip compression works on blocks of data and each block uses a single mode of compression. gzip is an appropriate candidate for its good compression and decompression rate, and high compression ratio. Even though the Deflate is the algorithm supporting compressing data in flexible ways, which are no compression for already compressed data and compressing with LZ77 and Huffman coding, gzip always performs compression even if the compressed file is slightly expanded. Fortunately, gzip only involves a few bytes expansion for the gzip file header and 5 bytes every 32K block of data in the worst case. Besides, the 9 levels are used to adjust performance of gzip, among which -1 gives fastest speed, while -9 offers best compression ratio.

2.6.2 lzo

Another candidate based on LZ77 in our experiment is lzo. It uses LZ77 with a hash table to perform searches. lzo is implemented in ANSI C and portable in different platforms. Its purpose is to provide a real-time compression program, therefore offers fast compression and decompression speed. It also means that it favors speed over compression ratio. Additionally, lzo supports the following features:

- Requires 64 kB memory for compression and no memory for decompression.

- Allows to slow down compression speed for a competitive compression ratio, while the speed of decompressing is not reduced.
- Includes compression levels for generating pre-compressed data which achieve a quite competitive compression ratio.

As mentioned in the documentation of lzo homepage, the algorithms supported are LZO1, LZO1A, LZO1B, LZO1C, LZO1F, LZO1X, LZO1Y and LZO1Z, and 37 compression levels are supported because the author wants to support unlimited backward compatibility. According to the specification (lzo homepage. 2009), “The naming convention of the various algorithms goes LZOxx-N, where N is the compression level. Range 1-9 indicates the fast standard levels using 64 kB memory for compression. Level 99 offers better compression at the cost of more memory (256 kB), and is still reasonably fast. Level 999 achieves nearly optimal compression - but it is slow and uses much memory, and is mainly intended for generating pre-compressed data.”

2.6.3 lzma

Since large variants of LZ77exist, we performed experiments on one more tool named lzma in the family. lzma compresses data using an improved version of LZ77 algorithm named Lempel-Ziv-Markov chain-Algorithm (LZMA), which combines LZ77 and DEFLATE compression algorithm. lzma is also implemented in the 7-Zip program (lzma homepage. 2009) and provides a high compression ratio and very fast decompression with small memory requirement. Same as gzip, it has 9 compression levels as well, -1 gives fastest speed and -9 offers best compression ratio.

2.6.4 ncompress

ncompress is based on Lempel-Ziv-Welch (LZW) algorithm. As mentioned in Section 2.3.2, LZW maintains a string translation table from the text being compressed instead of using sliding window in LZ77. lzw does not offer highest compression ratio instead it is one of the fastest compression program.

2.6.5 lzpxj

We have two tools based on combined algorithms of Lempel-Ziv-based compression and PPM. The one, lzpxj, supports good compression ratio, fast compression and decompression speed, and has one option (1-9) which select memory usage. Correspondingly, the minimum usage is selected by compression level 1, the maximum is 9 and the default is 6.

2.6.6 flzp

flzp is the other compression program using combination of LZ77-style string matching and PPM-style context modeling. As a fast and memory conservative compression program, it is used as a preprocessor to a low-order compression program to improve both compression ratio and speed. Additionally, the decompression speed is twice as fast as the compression speed.

2.6.7 bzip2

Developed by Julian Seward and released in July, 1996, bzip2 is a popular and stable compression program using the Burrows-Wheeler block-sorting text compression algorithm to convert sequences into strings of identical symbols, move-to-front transform and Huffman coding.

bzip2 generally offers better compression ratio than that achieved by LZ77 and LZ78-based compression programs, and approaches the performance of the PPM family of compression programs. bzip2 provides higher compression ratio for more files than gzip, however, more computation is the tradeoff. In this sense, bzip2 is slow at compressing, but as an asymmetric scheme, it has a relatively fast decompression.

Compression levels of bzip2 are similar to gzip, it has 9 levels from -1, the fast one, to -9, the best one. The levels set the block size from 100 k to 900 k when compressing, while the levels do not effect decompression (bzip2 homepage. 2009).

2.6.8 srank

The srank is also a member in the predictive compression family. It uses symbol ranking in BWT and is designed for compression speed rather than good compression ratio. It is designed for fast compression speed rather than high compression ratio. Besides, srank supports 8 compression levels and the maximum number of context is selected by option –C9 offering the highest compression ratio.

2.6.9 paq9a

The last software we evaluated is paq9a which was developed by Matt Mahoney and belonged to paq series. Paq9a is based on context mixing algorithm, it uses LZP preprocessing to speed compression of redundant data and is encoded by arithmetic coding. The option -1 of paq9a selects minimum memory but it is still extremely CPU and memory intensive.

2.7 Summary

In this chapter we began our exploration of data compression techniques with a description of lossless and lossy compression, also looked at three compression categories, namely statistical, dictionary and predictive compression in which the knowledge of the creative methods and their variants can be used to provide compression. Due to the many compression algorithms, the methods introduced in this chapter are chosen based on the algorithms contained in compression programs used in our experiment.

Table 2.1 Summary of the compression programs

Tools	Version	Algorithms	Levels
gzip	1.3.3	LZ77 + Huffman coding	1-9
lzo	2.03	LZ77	37 levels
lzma	4.32.7	LZMA	1-9
ncompress	4.2.4.2. orig	LZW	default
lzpxj	1.2h	LZP + PPM	1
flzp	v1	LZ77+ PPM	default
srank	1.1	Symbol Ranking in BWT	1-8
bzip2	1.0.5	BWT + Huffman coding	1-9
paq	9a	Context Mixing	not evaluate further

Besides, we have introduced 9 compression programs in Section 2.6 that give an overview of the compression programs evaluated in this study. One thing worth mentioning here is that lzpxj is tested only with compression level 1 because lzpxj with high compression level option is considered as a highly time-consuming compression program according to our initial experiemtns. Besides, paq9a is extremely CPU, memory intensive, even through with only the option

-1 of paq9a to be used, it still consumes unacceptable long time and correspondingly significant energy. So, it will not be evaluated further in our study. Table 2.1 summaries the aforementioned software along with their version, algorithms and the compression levels used in our experiments.

Chapter 3 EXPERIMENTATION SETUP

Our experiments aim to investigate the energy consumption of compression schemes and form a compression benchmark based on the results for further study. In this study, we not only try to examine consumed energy of different compression programs on Nokia N810, but also investigate the effects of transmission speed on energy consumption under different bitrates to fully understand overall energy consumption of the system.

In order to achieve clear understanding of the effects of compression on Nokia N810, a number of compression programs and plenty of compression levels will be examined. For this reason, it is necessary to have a measurement design for quick and easy acquirement of the energy consumption. In the following sections, the facilities used in our experiments, measurement procedures for energy consumption and bit rates will be introduced, and compressed files will be listed as well.

3.1 Measurement Equipment and Software

Nokia N810

Nokia N810, alias Internet Tablet, is a device used in this study to perform data compression and file transmission. It does not support mobile communication, but in instead allows Internet connection via IEEE802.11 b/g or Bluetooth. The specification of the device is described as follow: 400MHz TI OMAP 2420 (Open Multimedia Application Platform), 128 MB random access memory (N810 specification. 2007). One of the reasons we chose N810 as our measurement equipment is that Maemo Linux OS running on the device offers wide choices of open source compression programs and the device represents quite well a modern mobile phone.

NI USB-9162 and cRIO-9215

National Instruments USB-9162 is a single module carrier providing a highly portable solution for NI C Series modules. In our experiments, NI cRIO-9215 is used to capture voltage fluctuations. It features four differential analog input channels with a maximum single-channel sampling rate of 100 kHz. With plug-and-play connectivity via USB and NI-DAQmx software, the combination of USB-9162 and cRIO-9215 provides an ability to acquire, analyze, and present data in real-time for fast setup and quick measurements (National Instruments).

Jugi's Traffic Generator(jtg)

Developed by Jukka Manner in C language, Jugi's Traffic Generator (Jukka Manner. 2006) is a Linux traffic generator used to send and receive UDP or TCP packets. jtg supports a number of parameters to generate customized traffic for certain purposes and provides detailed summary about the traffic statistics. It is worth mentioning that the accuracy of the transmission is suffered due to Linux kernel itself and the effect from other running processes. However, jtg provides two methods to control packet sending interval. One is to use select() function in which Linux kernel only provides an accuracy of around 10 milliseconds. The other is to use busy-waiting, it forces the sender to use sleep() function to control the interval, however it may occupy all CPU usage.

3.2 Energy Measurement Setup

To achieve uninterrupted energy consumption measurements, it is necessary that the voltage input to the Nokia N810 remains stable. This will ensure more accurate measurements results on battery powered devices. The battery of the N810 is replaced by a battery adaptor which connects to a 3.9 V DC power supply.

Power consumption is acquired by calculating by the following equation:

$$P = U * I$$

where U is the electrical voltage on the N810 measured with the NI data logger, I is the current.

The parameter to be measured of the N810 will be the voltage, since NI data logger reporting the voltage fluctuations is probably the simplest way to calculate the power consumption of the device. Fig 3.1 illustrates the setup of measuring the voltage of the N810. The battery of the N810 is replaced by the battery adaptor, which connects serially to a 0.1 Ohm resistor. The voltage readings are recorded with a rate of 1000 samples per second across the resistor using an additional Windows PC with NI-DAQmx software running on.

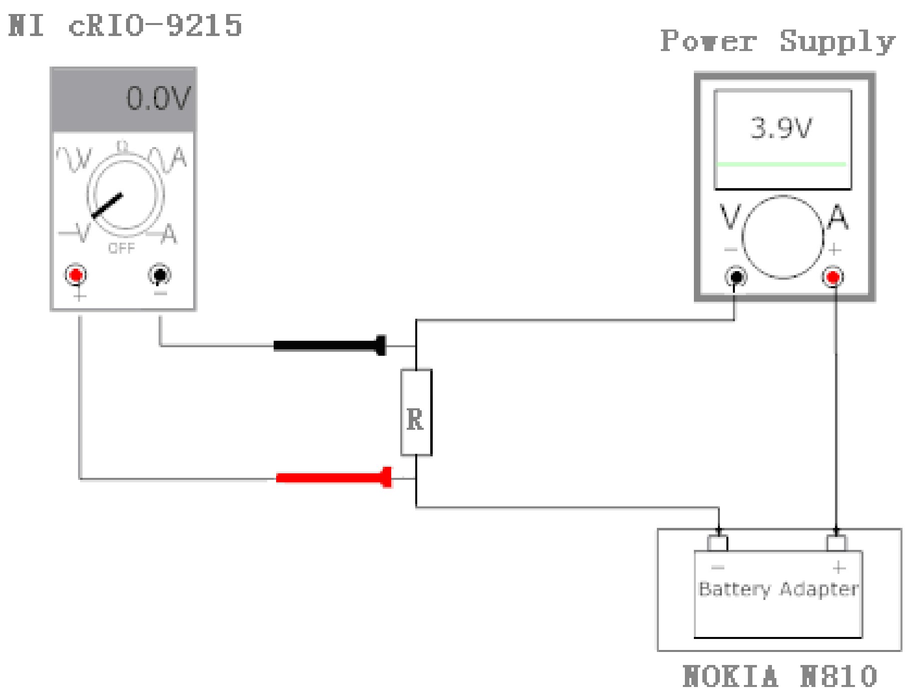

Fig 3.1 Energy measurement setup

Furthermore, the equation P = U * I is extended to P = $U_{0.1\Omega}/0.1 * (3.9 - U)$

where $U_{0.1\Omega}$ is the voltage on the resistor.

Correspondingly, energy consumption is acquired by calculating E = P * T.
where T is the running time for a process, such as running time for compression, decompression or transmission.

Depending on the compression programs, the duration of compressing and decompressing takes from less than one second to several minutes or even longer for certain file types. Therefore, synchronization is needed for accurate measurements. Optimally, synchronization between the PC with NI-DAQmx software and the N810 is done before measurement using NTP or scripts. Fig 3.2 illustrates an example of voltage fluctuations of the N810 when bzip2 performs compressing and decompressing. As observed, basic voltage requirement of the N810 is around 3.8575 V, while voltage curve becomes unstable when compression starts.

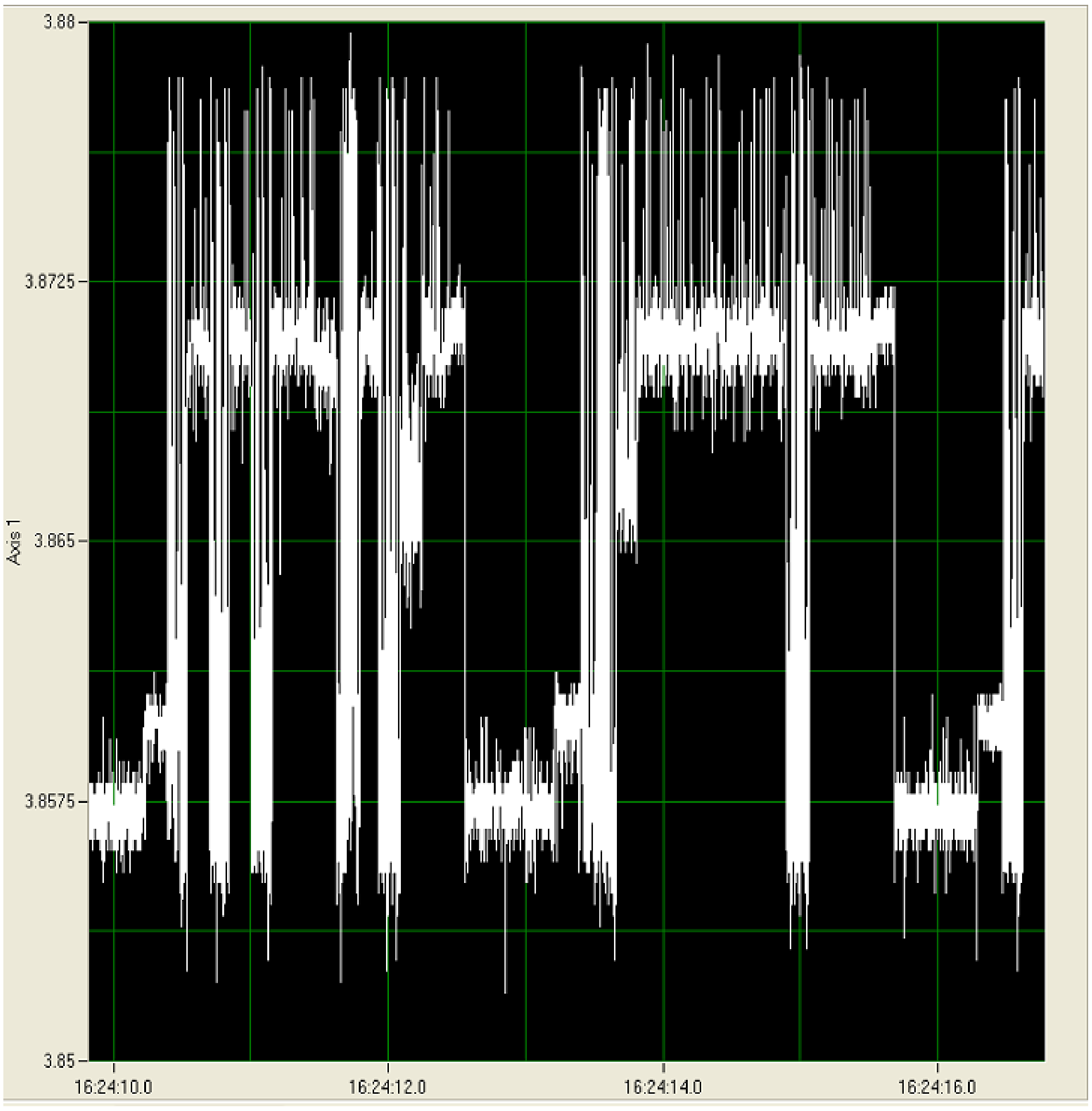

Fig 3.2 Voltage fluctuations in real time

3.3 Bit Rate Measurement Setup

In order to investigate bit rates' effects on energy consumption, our wireless link experimental equipments used in this study are the N810 and a desktop computer with Ubuntu 8.04 and a D-Link IEEE 802.11b/g wireless USB card installed running in ad-hoc mode.

The packet sending intervals are adjusted by jtg, namely bit rates are controlled. In order to avoid negative effects on measurement from high CPU usage, select() function is used in our experiment to adjust packet sending interval. In our setup, the maximum sync up speed is 11 Mbps and real speed is around 1448 Kbps when sending UDP packets with 1400 bytes packet length. With implemented report features, jtg gives the summaries of the traffic and the basic statistical results including necessary information needed in the analysis.

Similarly to the energy measurements of compression and decompression, voltage values logged by the data logger and transmission time are used to calculate energy consumption. With sending and receiving time and bytes, we can also obtain the energy consumed on a bit or a second at sender and receiver. In detail, the energy consumed on sending is measured on the N810 when it sends packets to the PC. In contrast, receiving energy is obtained also on the N810 with a difference that packets are sent from the PC.

3.4 Test Files

Web surfing and media files downloading constitute main data traffic on mobile devices, thus our experiments are mainly focus on typical file types, such as web pages, media files and binary machine files. The details of the files we examine are listed in Table 3.1 and file names and sizes are given in two columns.

The compression levels for each compression program aforementioned in Table 2.1 are used to examine each file. The higher levels of gzip, bzip2 and lzma provide higher compression ratio and slower compression speed. The compression programs lzpxj, fpaq and paq achieve very good compression ratio at the expense of insanely long runtime and enormous consumption of energy correspondingly. Especially, paq takes excessive time, thus it is not considered in the following chapters.

Table 3.2Test files

File names	Size(B)	File names	Size(B)
A10.jpg	842468	mean.wma	1462495
sample.html	1058244	nb.swf	3861613
Flash.pdf	4526946	qq.exe	842468
Heart.mp3	2313950	rafale.bmp	4149414
mcmd.bin	2760821	xslspec.xml	1584495

Chapter 4 TRANMISSION IMPACT

In order to use compression to achieve energy efficiency and energy awareness in future Internet communication on hand-held devices, it is necessary to acquire a good understanding of the energy consumption behaviors of wireless interfaces. The knowledge helps us to obtain the total energy consumption of a compression-capable communication in either uplink or downlink. Besides, it is also useful to guide future research and implementations.

As mentioned in Section 3.2, the measurements are performed in ad hoc mode to exclude the effects of the packets transmitted in infrastructure networks, such as broadcast traffic and access control messages. This chapter describes the details of our energy consumption measurements on transmission in IEEE 802.11 wireless networks. Power consumption is one of the metrics to be examined and energy consumed on sending and receiving a bit is also helpful to analyze energy consumption pattern of the device. Clearly, packet sizes surely affects traffic load, so the energy consumption in sending and receiving data packets of various sizes is also one of the study topics. Last but not least, bit rate is one of the most important metrics to evaluate wireless networks. Therefore it is necessary to investigate the impacts of bit rate on transmitting packets over radio for energy-aware communication.

Here, the results of basic energy requirements of the N810 are given first as a baseline to evaluate the following works. The power consumption is around 0.0149 watts when radio is turned off and the value is 0.83448 watts when radio is on. The values were obtained under conditions that the screen of the device was turned off, no application was running at the same time and power saving was also off. Besides, all measurements in the thesis are carried under same circumstances. In the following sections, two transmission factors are about to be discussed considering energy consumption.

4.1 Packet Sizes Impact

This section addresses the question about the effects of packet sizes on energy consumption on the Nokia N810. We want to understand the variations of energy consumption when increasing the size of the packet to be transmitted, and examine the behavior of energy consumption when transmitting UDP packets simply because of avoiding TCP retransmissions. The answer to the question could help us to obtain a basic understanding for finding optimum packet sizes in wireless networks and making right decisions when we implement energy-aware communication in future study.

We have used a number of packet sizes in the experiments ranging from small packet with 50 bytes to large packet with 1400 bytes and the transmission rate is controlled by setting packet sending interval to 100 ms. Power management can not improve energy efficiency, but it helps to save energy because radio is turned into sleeping state while there is no traffic. Therefore, the experiments on transmitting different-sized packets are without power saving.

First of all, sending requires a bit more power than just receiving and the energy needed to send a bit is also higher. This observation shows that sending packets demands higher energy than receiving. More details are presented in Fig 4.1 and Fig 4.2, in which energy consumption of sending and receiving UDP packets with different packet sizes is illustrated. The energy is showed in two ways, namely consumed energy per second and the energy per bit using square and dotted lines respectively. Considering relative small scales of the power consumption in both figures, the initial points are not marked from zero in order to give a better illustration of the energy consumed per seconds.

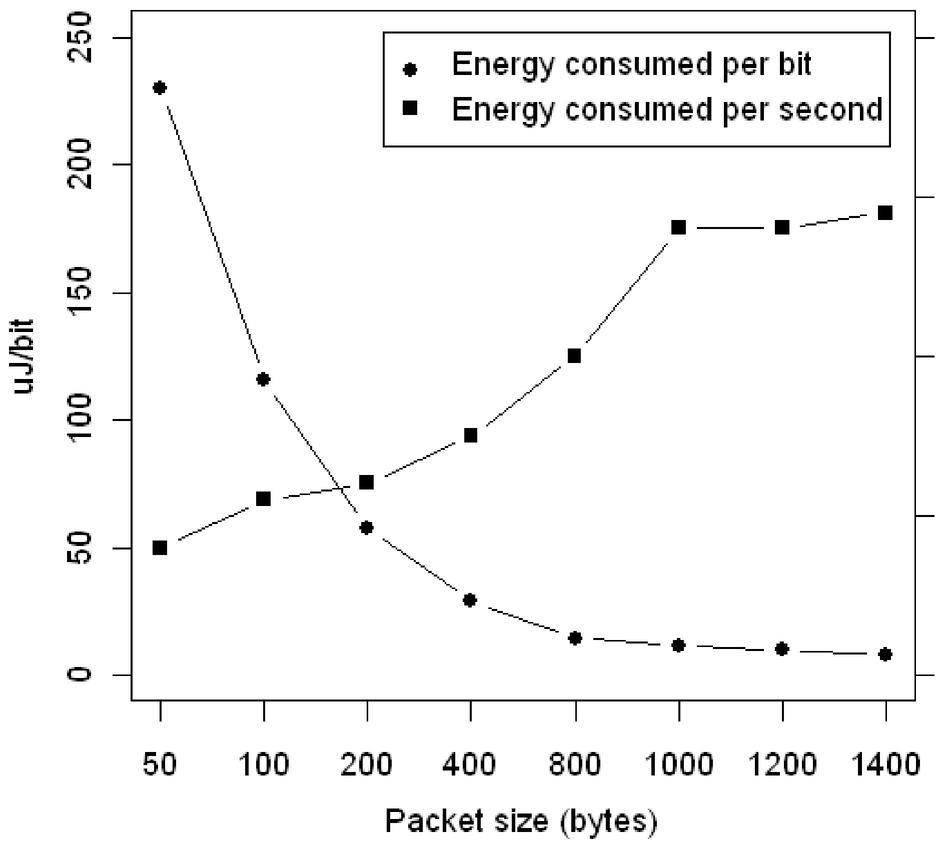

Fig 4.3 Consumed energy on packets with different packet lengths when sending

Fig 4.1 shows that the packet sizes show different trends of impacts on the power level variation for each presenting way. Energy consumed per second on sending packets of various sizes is presented as a linear increment along the packet sizes. However, the basic power consumption of radio is so high that the trend is not negligible. In contrast, a strong impact on energy consumed per bit can be observed. The energy consumption dramatically drops as the packet size increases. Moreover, the consumed energy roughly falls down to half when the packet size doubles. As clearly shown in the figure, power consumption does not have an obvious change even when the packet size enlarges from 50 bytes to 1400 bytes, thus per-bit energy consumption is highly depend on the size of the packet to be transmitted.

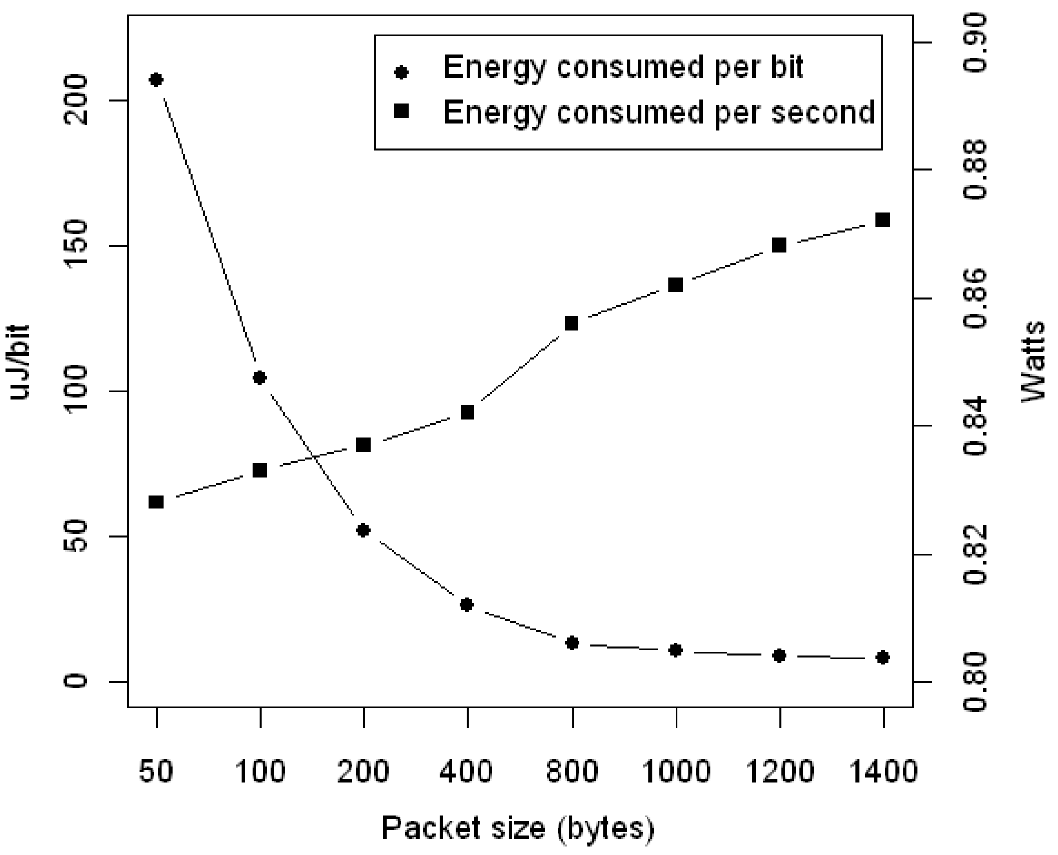

Fig 4.4 Consumed energy on packets with different packet length when receiving

Energy consumption associated with receiving a bit is similar to the case of sending it. Fig 4.2 also shows that the basic consumed power cost of receiving is relatively high, while the incremental cost of data is relatively low. Therefore, energy consumed on receiving a large packet is low, and shows that the consumed energy on a bit drops to 8.657 micro joules.

4.2 Transmission Rate Impact

As observed in Section 4.1, sending and receiving small packets is extremely inefficient in the of view of energy consumption per bit. In order to maximize energy efficiency and show the benefits of compression in wireless communications, 1400 bytes are used as packet size in our experiments. Therefore, the tool jtg is configured to send 1400 bytes UDP packets at different intervals. The select() function is used as sending interval control instead of busy waiting to avoid high CPU usage and unnecessary power consumption resulting in inaccuracies. At first, the jtg is in sending mode to generate traffic load at given intervals starting from 0 ms with a increment from 50 ms to 300 ms. The energy costs of sending are illustrated in Fig 4.3. Based on the measured energy of the transmission, the numbers of packets and transmission time, the results are demonstrated in the same way as in Fig 4.1, showed in two kinds of y-axes.

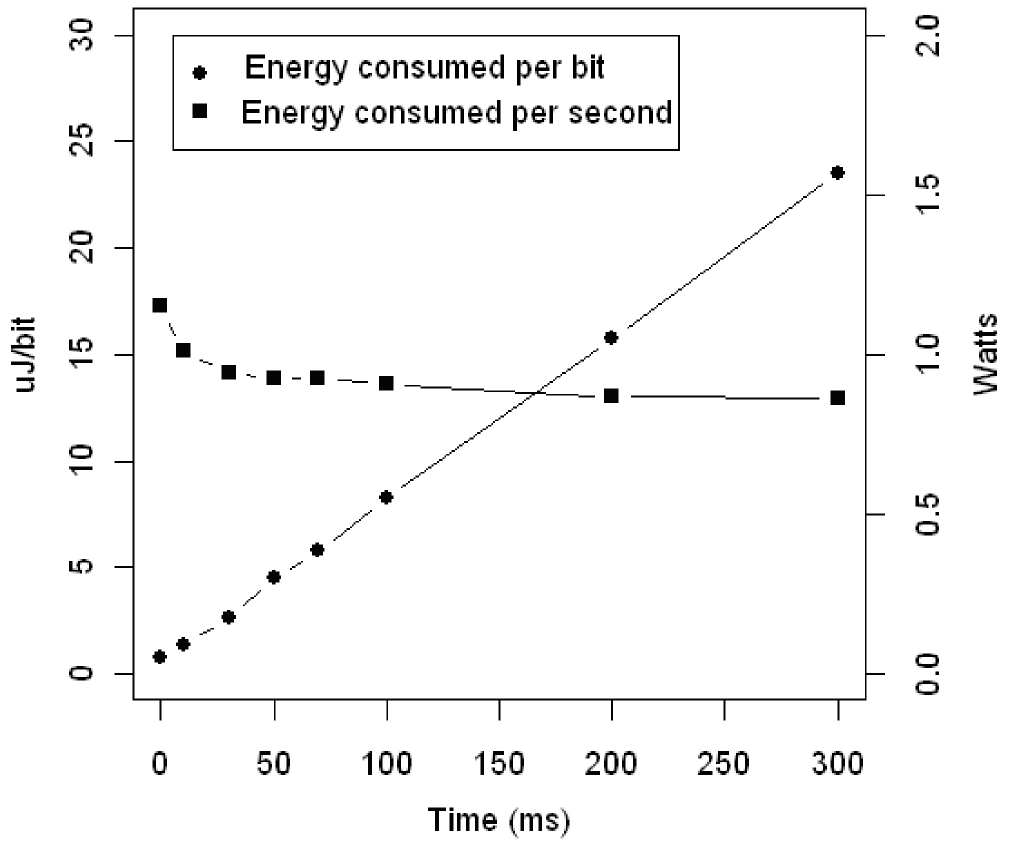

Fig 4.5 Consumed energy under different transmission intervals when sending

In Fig 4.3, the energy consumption per second only changes slightly. Compared to the cost of sending, the high energy requirement of just turning the radio on is already excessively. Thus, the incremental energy required for faster sending has limited impact on power consumption. While the energy consumed on a bit is roughly linear proportional to the sending intervals due to linear decrease in the number of the bit to be transmitted.

The x-axes can be represented in another way by converting the expression of packet sending intervals to the transmission rate. The corresponding dots to the ones in Fig 4.3 are illustrated in Fig 4.4, expressing transmission rate impacts on energy consumption. With the adaption of packet sending intervals, the transmission rates range from 37.50kb/s to 1448.67kb/s. The curves reflect the same knowledge expressed in Fig 4.3 that fast sending reduces energy consumption on a bit, while the power consumption is only slightly increased due to the high fixed overhead.

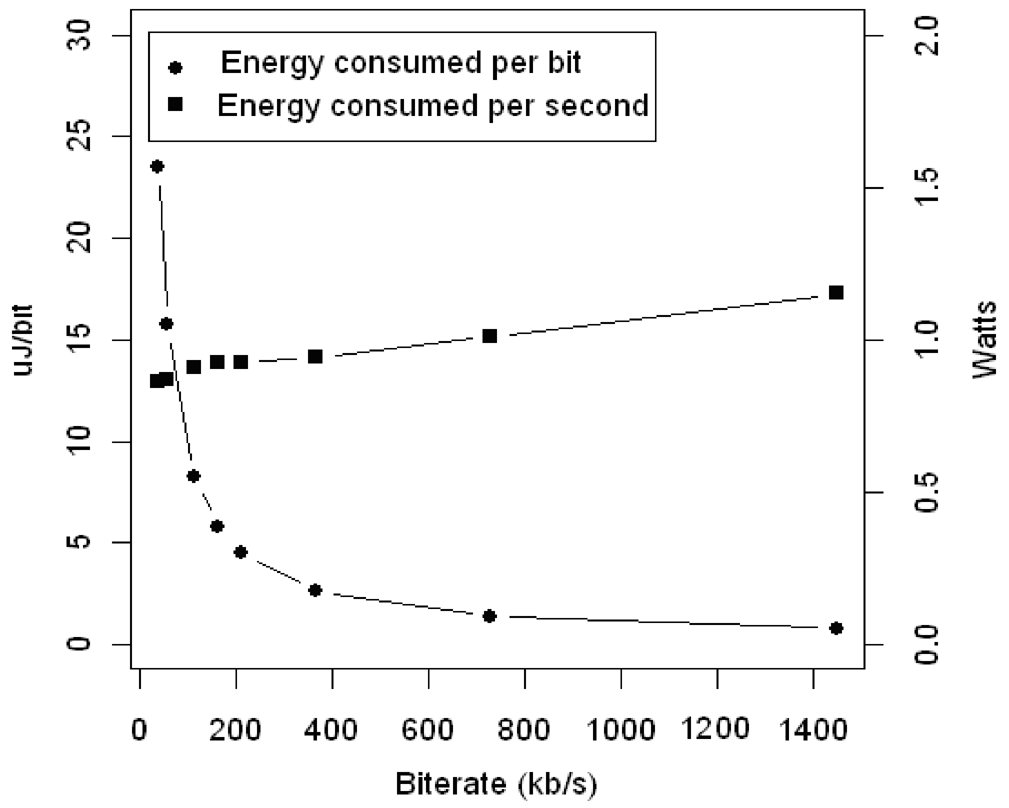

Fig 4.6 Consumed energy under different transmission rates when sending

Regarding receiving packets, the power consumption and the consumed energy associated with it is presented in Fig 4.5 and Fig 4.6. The consumed energy on receiving shows a similar trend to the case in sending except that the overall energy consumption is relative less.

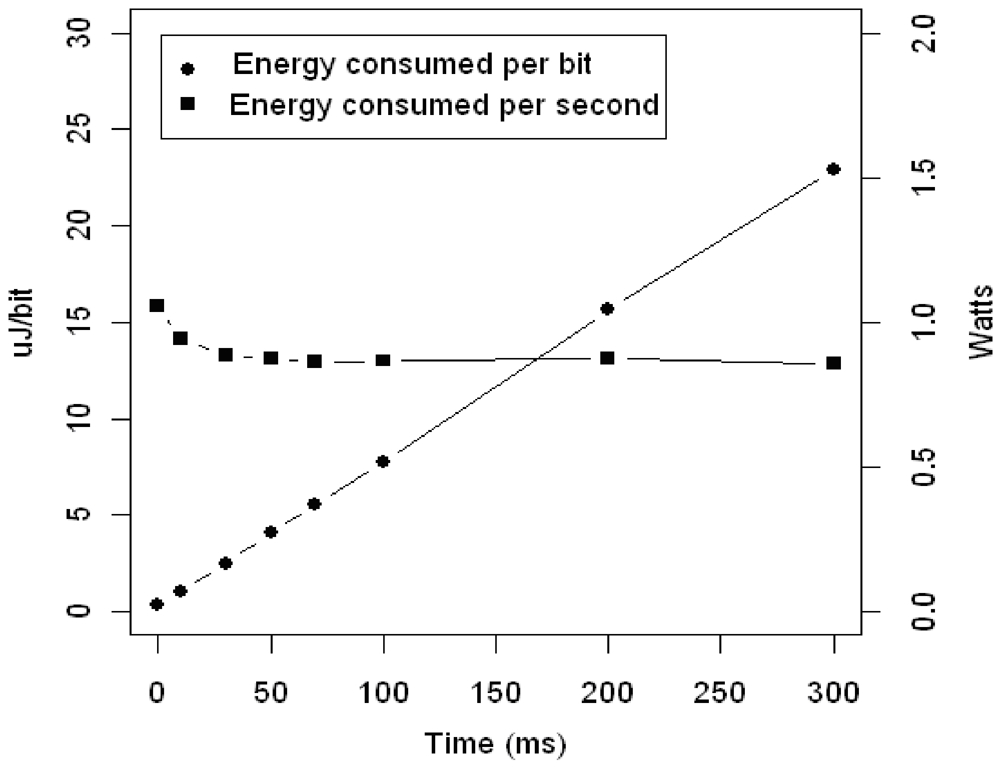

Fig 4.7 Consumed energy under different transmission intervals when receiving

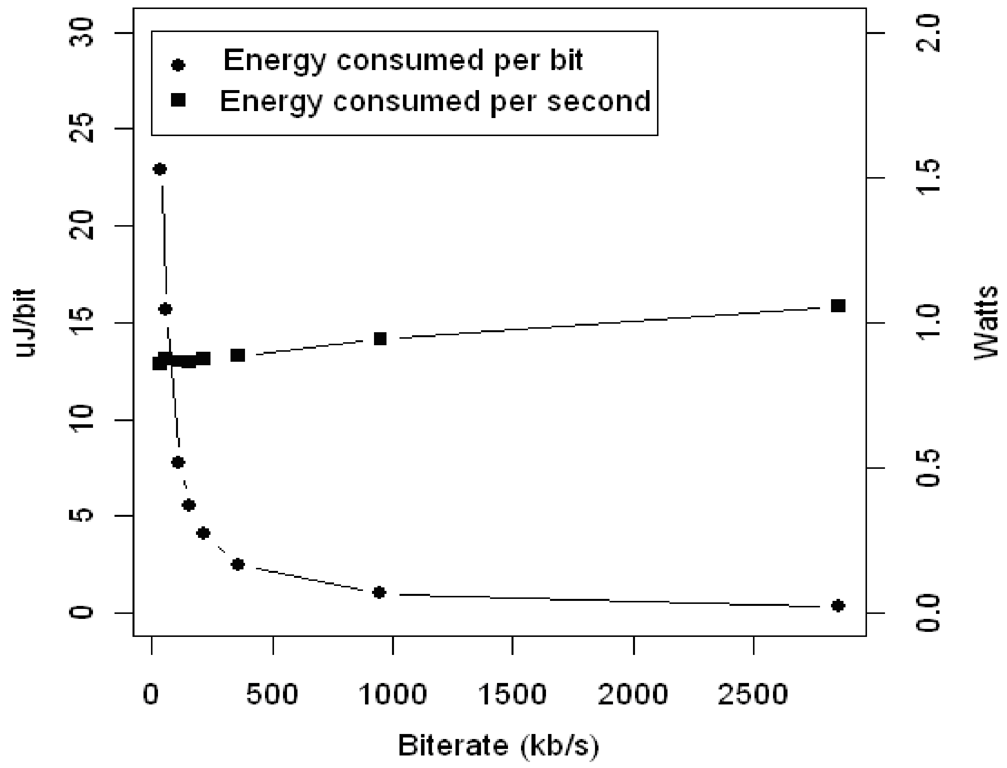

Fig 4.8 Consumed energy under different transmission rates when receiving

4.3 Summary

During series of measurements of transmission on the N810, the observations show that the energy consumption of transmission could be formulated as a function of the number of bits to be transmitted plus the overhead. Moreover, transmitting small packets or sending with slow transmission rate have disproportionately high energy costs on a bit, while energy consumed per second does not show fluctuations because radio channel acquisition overhead is large but the energy consumption on transmitting bits is relative low. The phenomenon could be interpreted as: regarding a certain mount of data, sending large packets over a fast link is energy-efficient, or in other words that it helps to reduce energy consumption when sending the data in a burst way and then force the radio card to a sleep state. The possible tradeoffs involved are an increase in power consumption and congestion in the link.

The idea to save energy is to maximize the time that the radio spends in sleep mode. Considering characteristics of compression, it is helpful to shorten transmission time resulting in reduction of consumption of energy resource. The following chapter will discuss the impacts of transmission rate and file types when using compression in wireless networks.

Chapter 5 COMPRESSION IMPACT

David Salomon defined compression as “Data compression is the process of converting an input data stream (the source stream or the original raw data) into another data stream (the output, or the compressed stream) that has a smaller size. A stream is either a file or a buffer in memory.” The reasons for data compression getting popular, as mentioned in his book “Data Compression: The Complete Reference”, is to save storage space and shorten transmission time. His opinion indicates that the design and normal usages incorporated with data compression are not energy-oriented. Thus, blind or unconditional compressions for energy-aware communication related to wireless networks may result in wasting of energy and even slowing down transmission rate.

This chapter aims to carry an experimental study on deploying compression for energy-aware communication. In the of view of energy savings, energy consumption is the primary metric to evaluate the compression programs instead of only focusing on such traditional metrics as compression ratio and speed. We evaluate a set of lossless compression programs with different compression levels described in Section 2.6. As known, many kinds of data are not suitable for compression because of a poor compression ratio. In order to obtain a good understanding of the behavior of energy consumption on different type of data, ten kinds of data listed in Table 3.1 are examined. In such circumstances, not only the downstream and upstream energy impacts will be examined, but also we consider overall energy consumption to see how different compression can lead to energy savings for the whole system.

Firstly, we give an overview of the power consumption and energy usage behavior on the N810. The data of compression/decompression time and the energy consumption for every compression program on each file are plotted in Fig 5.1, in which a summary of power consumption is given in box plots. As illustrated, the power needed for compressing centralizes in the range from 0.34350 watts to 0.58350 watts with 0.44620 watts as mean. While compression program needs 0.32340 watts on average to decompress data and corresponding power consumption range is 0.25460 watts to 0.37510 watts.

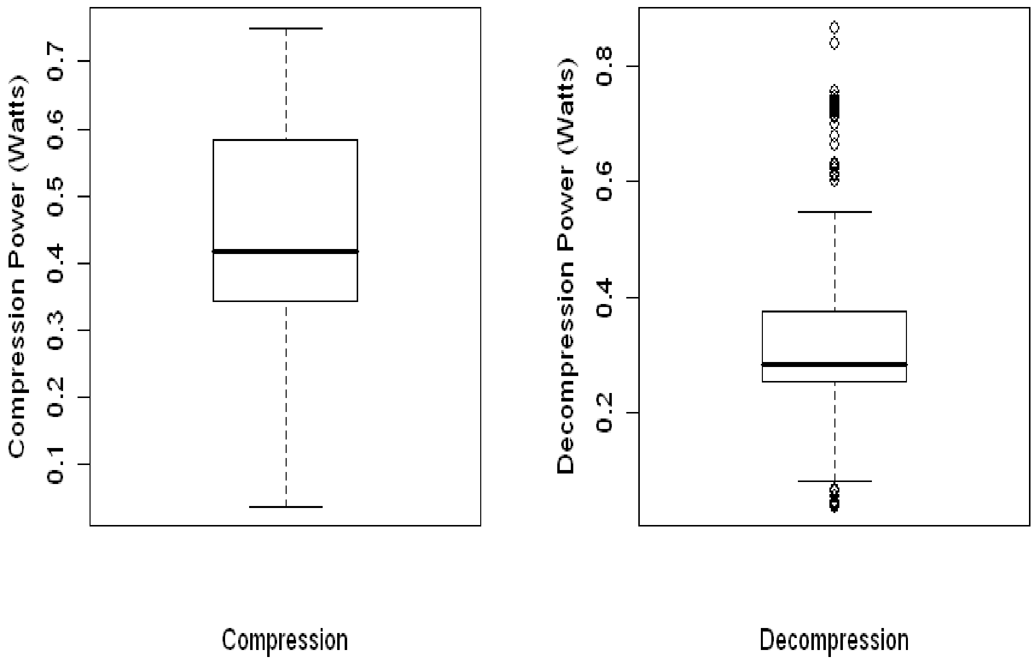

Fig 5.9 Summary of power consumption

Basically, energy consumption of compression and decompression is depended on CPU cycles, memory accesses and processing time. As mentioned earlier, some compression algorithms are highly CPU and memory intensive. Even so, processing time is the primary factor of affecting energy consumption. As explored in Fig 5.2, the measured values of compression time and energy consumption are drawn in scatter plot, in which a strong correlation between the two variables is presented. Stated more precisely, the figures show that there is an overlapping of most of the dots and the lines and no anomaly that is so high above the lines. The observation indicates that consumed energy on compression and decompression are highly dependent on their processing time.

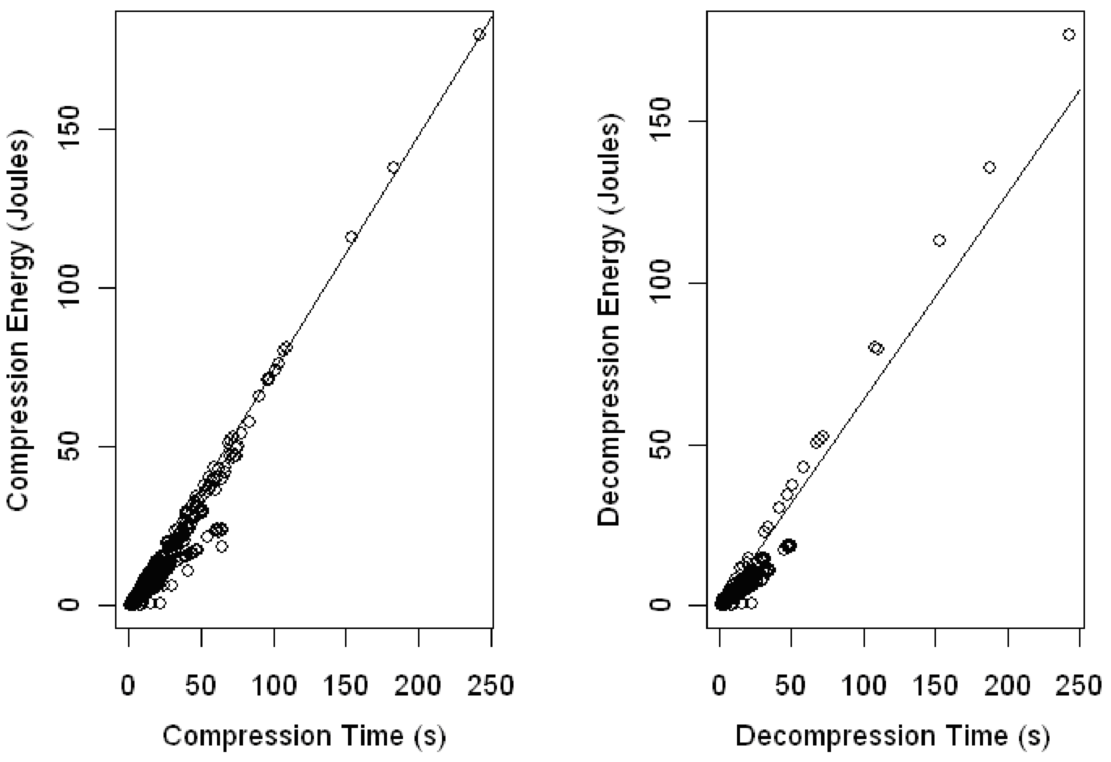

Fig 5.10 Time VS consumed energy when compressing and decompressing

Based on the type of the files, we will discuss the relationship between file content and wireless network status in the perspective of energy consumption. The analysis is presented in the way of figures in the following sub-sections and the detailed numbers used to draw the figures are categorized and listed in APPENDIX A.

5.1 Hard-to-compress Files

As known, Internet Point-to-Point Protocol (PPP) includes some compression methods, such as Magnalink Variable Resource Compression Algorithm (MVRCA) and a high performance LZ derivative named FZA (D. Rand. June 1996). Moreover, ITU-T V.42bis recommendation (V.42bis) is incorporated with a variant of the Lempel-Ziv-Welch (LZW) compression method (ITU-T. 2006-09-29). These communication protocols with integrated compression methods are used in most modems to compress streams on the physical layer.

Unfortunately, many kinds of data are not suitable for compression. Viewing from the perspective of file types, lossless data compression programs always fail to compress certain files if compression programs can not find repeating patterns inside files such as pre-compressed and already highly compressed data. Moreover, some compression programs generate even bigger size file of certain data while some others work well. The answer, of course, is that such files do not have much redundancy to remove. An example to show is to compress a file with totally random text, in which each symbol occurs with equal probability, thus same length is assigned to the represented codes based on the probability and this does not help to reduce the file size.

In our dataset, almost all the selected compression programs fail to compress the files with extension of JPG, MP3, WMA and EXE. Indeed, these files have been highly compressed already, as a consequence that none of compression programs can achieve good compression ratio. In Fig 5.5, the highest compression ratio of selected compression programs for each file and their compression time are presented in dotted lines and columns respectively. Cat is a Linux command to concatenate files and print on the standard output, which is used as direct memory copy to compare with other results in our experiments. As observed, flzp and ncompress result in an expansion for all the files, srank only offers a little saving in izes for JPG and MP3 files with certain compression levels, also the other compression programs give limited benefits for JPG, MP3 and WMA files. The file with EXE extension, the hardest file to be compressed, can be only slightly reduced in size by gzip and lzo.

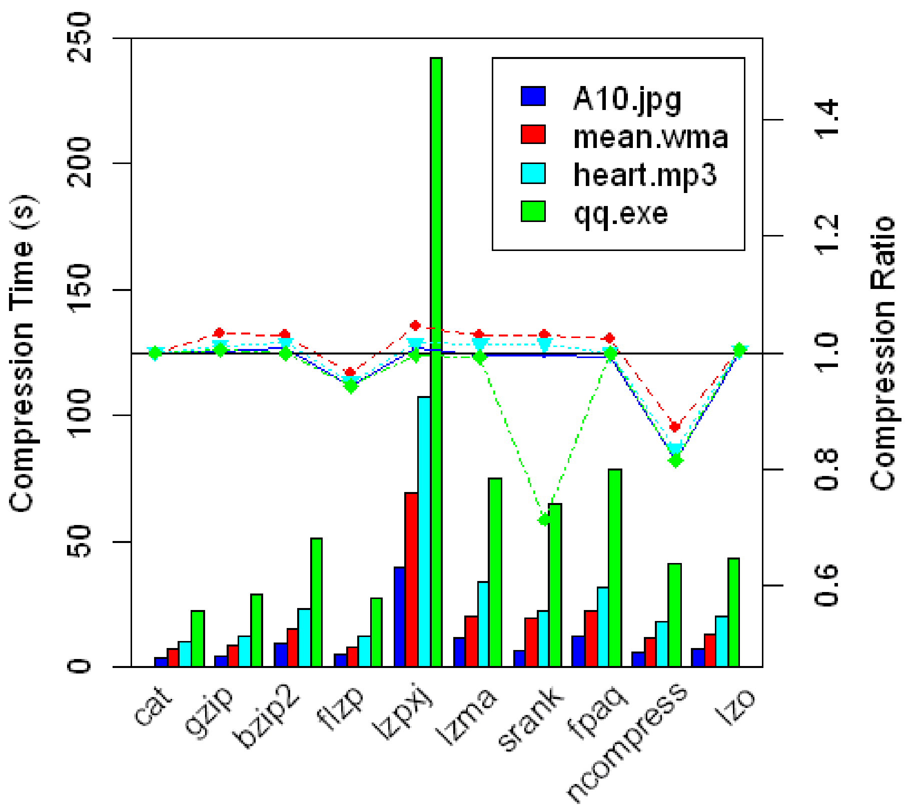

Fig 5.11 Compression ratio and time of different compression programs

The energy consumption of compressing, decompressing and transmitting the JPG, MP3, WMA and EXE files are illustrated in Fig 5.4 and Fig 5.5. According to the results in Section 4.2, the energy consumption is measured under the condition that the packet sending interval is 10 ms to achieve maximum energy savings. As observed, sending directly is the most energy efficiency way to transmit highly compressed files. Of course, the overall energy consumption is correspondingly increased when any compression methods are incorporated. Considering very limited energy saved (or wasted in some cases) from transmission of compressed data and the required energy consumed on compression, it is not worth applying compression to these files. Indeed, attempting to compress these already compressed is a waste of time and even increase energy consumption. Even if the resulting file size may shrink, the gain of energy savings is still negative due to excessive compressing and decompressing overhead.

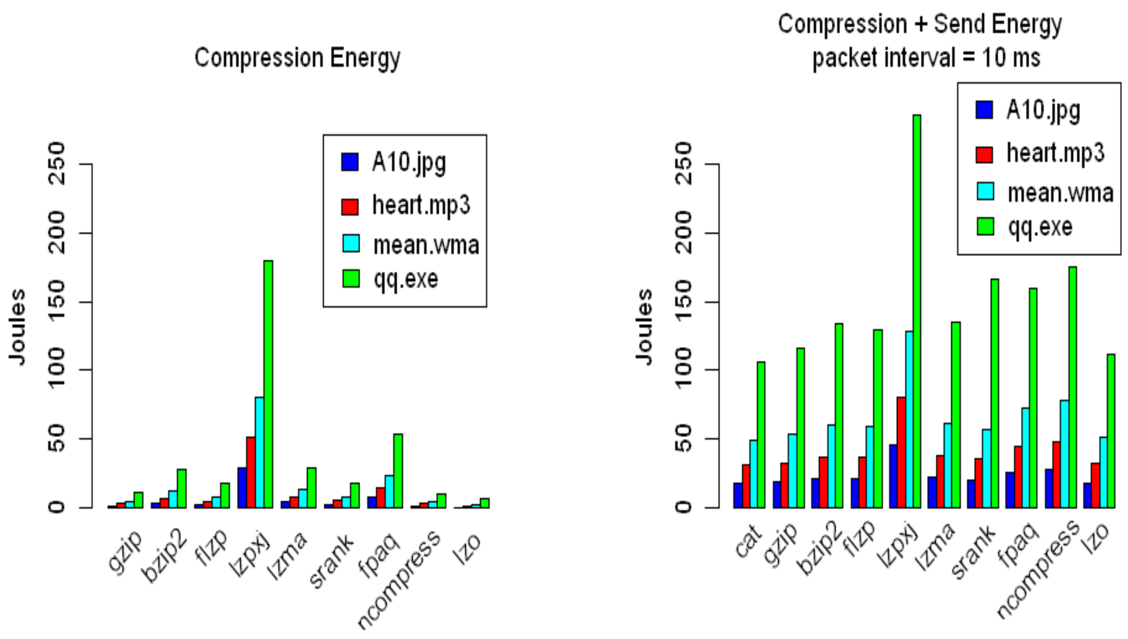

Fig 5.12 Energy required to compress and send JPG, MP3, WMA and EXE files

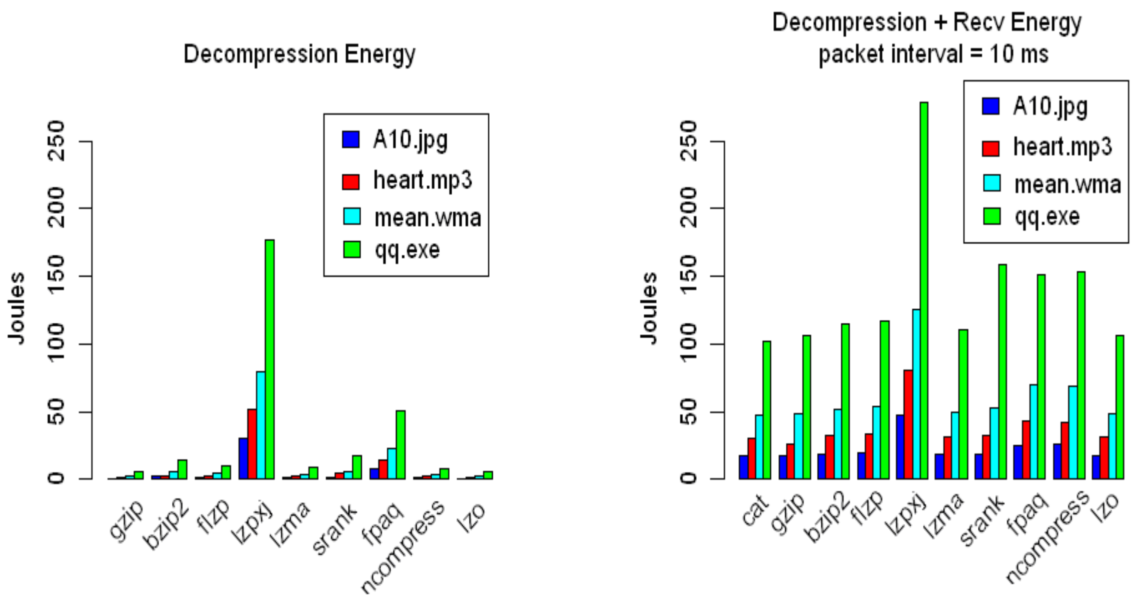

Fig 5.13 Energy required to receive and decompress JPG, MP3, WMA and EXE files

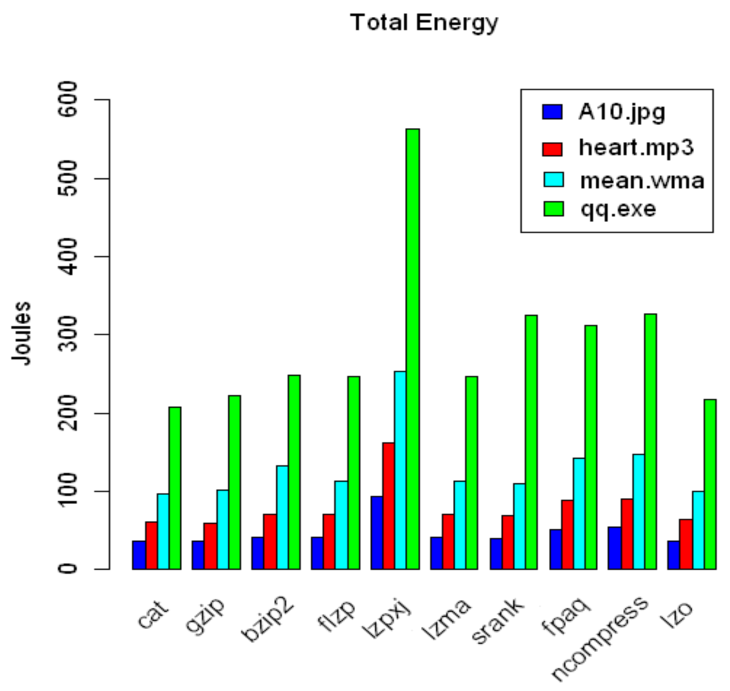

Fig 5.14 Total energy required to transmit JPG, MP3, WMA and EXE files

Fig 5.6 gives an illustration of overall energy consumption when transmitting hard-to-compress files. The total energy showed in the figure consists of the consumed energy on compressing/decompressing, sending and receiving. As observed, extra energy is involved by compression. Thus, it is energy saving to transmit these files directly through radio without any compression in the perspective of overall energy consumption.

5.2 Compressible Files

This sub-section discusses the rest files listed in Fig 5.7. Grounded upon our investigations, gzip, bzip2, lzpxj, lzma and lzo achieve better compression ratio than the others on the test files listed in Fig 5.7, however, lzpxj is highly time demanding to achieve the outstanding ratio as showed in the upper figure, also fpaq requires extreme long time to compress the files. Among the compression programs, gzip and lzo are two standouts in the performance of ratio/time meaning that the time cost on the ratio is less than the others.

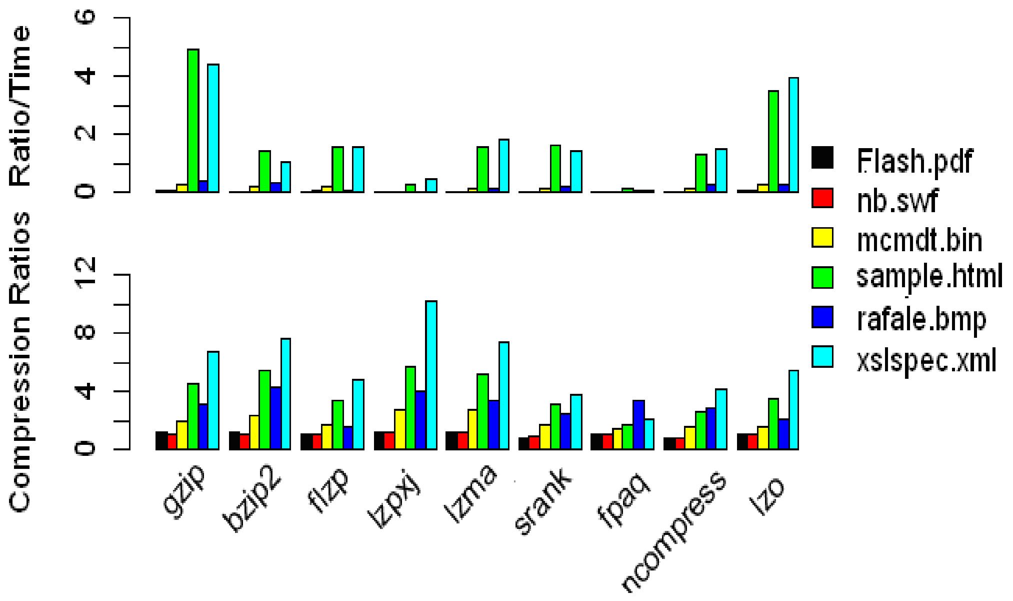

Fig 5.15 The best ratio/time of the compression programs and the corresponding ratio

PDF and SWF files are discussed first. These files, frequently delivered over Internet, are not only a better data structure but also efficiently compressed, thus all the compression programs perform unsatisfactorily. More specifically, a number of compression schemes are supported by PDF to reduce file sizes. For example, JPEG and JPEG2000 are used to compress color and grayscale images, CCITT (the facsimile standard, Group 3 or 4), Run Length Encoding and JBIG2 (Joint Bi-level Image Experts Group) are applied to compress monochrome images, LZW and Flate are used to compress text, graphics and images (Adobe Systems Incorporated. November 2006). As to SWF, it uses such technologies as elements reusing and data structure shaping to minimize file sizes, and also supports a variety of bitmap formats such as JPEG, ZLIB bitmaps, and uses ADPCM (Adaptive Differential Pulse Code Modulation), MP3 and Nellymoser for audio and video compression (Adobe Systems Incorporated. November 2008). Observations show that some compression programs can still offer better compression ratio than offered to aforementioned JPG, MP3, WMA and EXE files. Even through considerable time and energy are consumed to compress the files, which in return only saves little space, the limited gain in ratio from compressing PDF and SWF files offers an opportunity to achieve an energy-saving transmission under certain condition since the energy consumed on a single bit transmitted over wireless is costly, especially over slow links. The reason for the phenomenon is demonstrated in Fig 5.8, in which the least values of energy consumption achieved by certain levels are showed. Energy required for compressing and decompressing are considered as a basic demand illustrated in the left figures, moreover, transmission costs are taken into account under two conditions, namely a fast link sending packets at 10 ms interval and a relative slow link with a 30 ms sending interval. About 0.8056 micro joules energy is required on one bit with 10 ms sending interval and the corresponding energy required to receive one bit is 0.3941 micro joules. Whereas sending one bit demands 2.637 micro joules energy and the N810 needs 2.530 micro joules to receive the bit when sending interval is 30 ms. The big different energy requirement on transmission of a bit over the fast and slow links help to stress emphasis on using an adapted compression on PDF and SWF files. As showed in the middle figures in Fig 5.9 and Fig 5.10, using compression involves more energy consumption than directly sending the files when packets are sent at 10 ms interval, whereas, gzip and lzo save energy for both files when the N810 sends packets at 30 ms interval. The tools gzip, bzip2, lzma and lzo help to release the pain on transmitting the files at the receiver. Considering overall energy consumption showed in Fig 5.11, gzip, lzma and lzo are useful to transmit PDF and SWF files. In summary, it is even valuable for energy saving to compress such kinds of well-organized and compressed data as PDF and SWF files when radio links are suffered.

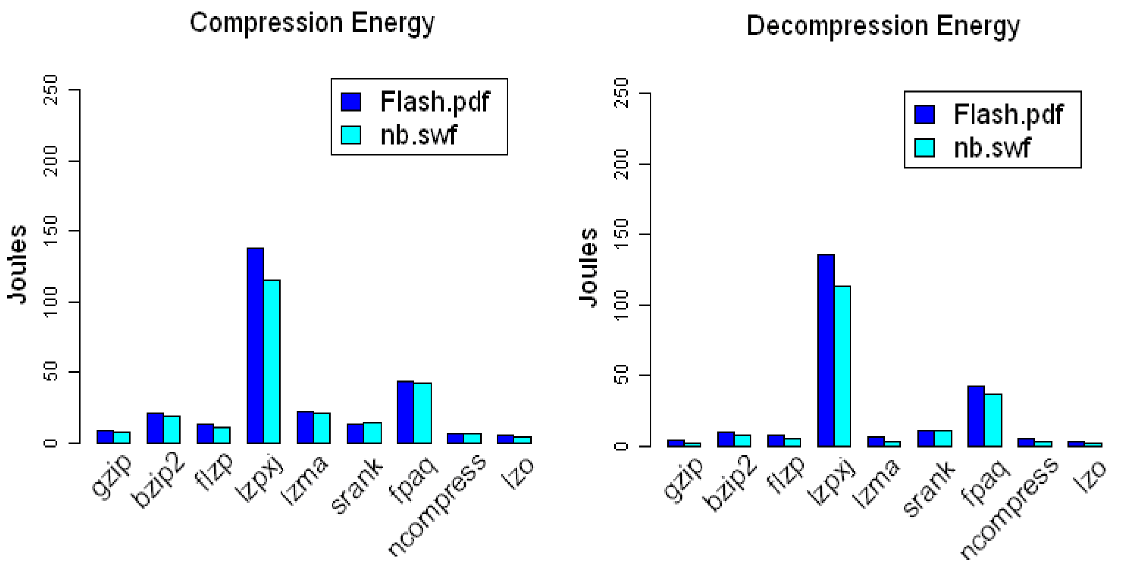

Fig 5.16 Energy required to compress and decompress PDF and SWF files

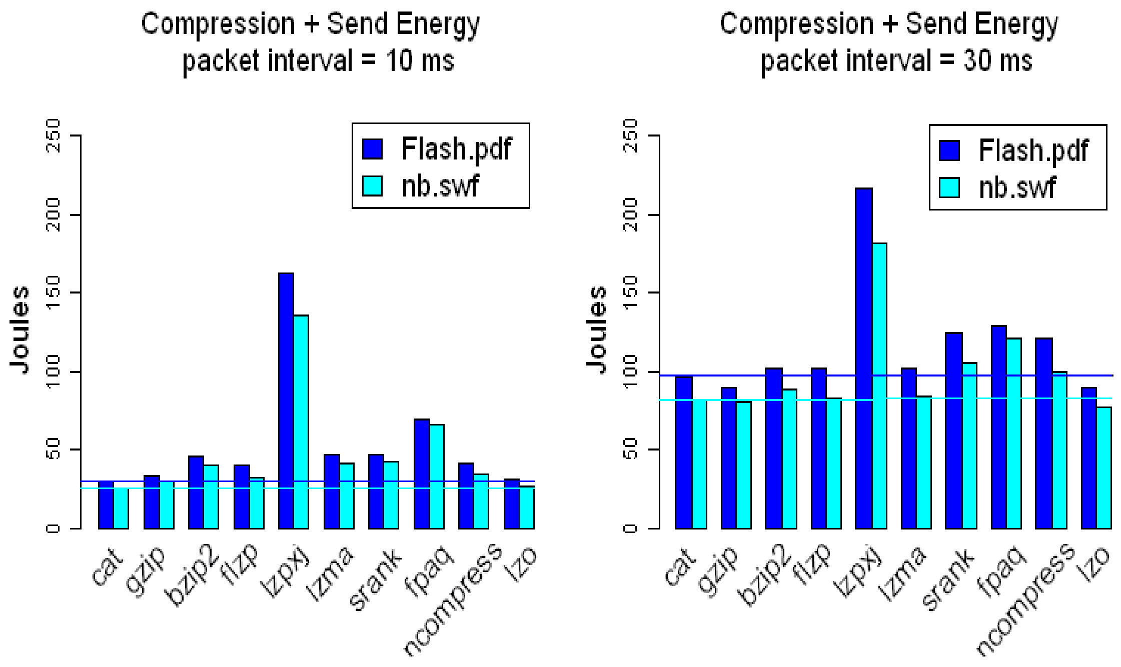

Fig 5.17 Energy required to compress and send PDF and SWF files

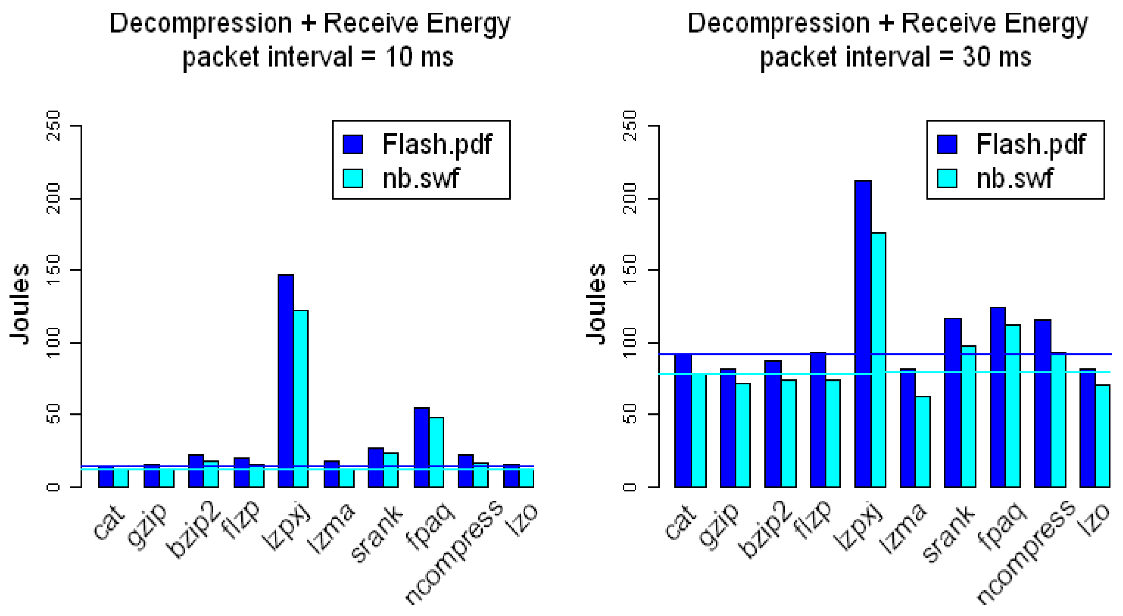

Fig 5.18 Energy required to receive and decompress PDF and SWF files

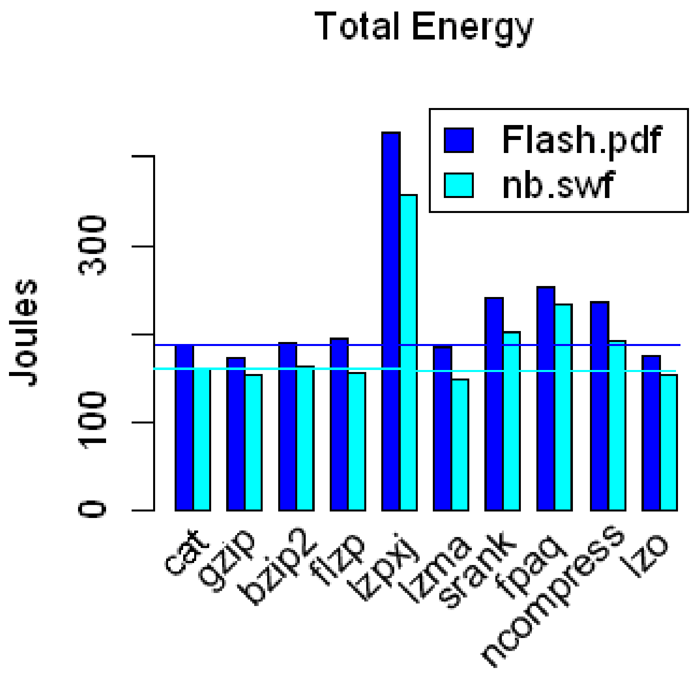

Fig 5.19 Total energy required to transmit PDF and SWF files

More specifically, when the network is slow or overcrowded, more time is worth spending on compression to trade for reduction in file sizes. On the other hand, if the network is fast and its utilization is low, it may be better to transmit such data as PDF and SWF files without any compression, saving the CPU time for other processes.

By contrast, the energy consumption of compressing, decompressing and transmitting BIN, HTML, BMP and XML files are showed in Fig 5.12, Fig 5.13 respectively. The files belong to easy-to-compress files and compression offers a possibility of energy savings even when packets are sent at 10 ms interval. Most of the compression programs provide an energy efficient transmission of the files, while lzpxj and fpaq demand extremely long time and energy to compress. Furthermore, gzip, a good competitor in energy-saving to send of BIN, HTML and XML files, reduces the energy consumed on sending the files to 68%, 23% and 18% of the energy needed for just sending directly. To compress and send BMP file, ncompress gives a better performance of 56% saved energy than 50% saving offered by gzip. At receiver, similar energy is required by all the compression programs expect lzpxj and fpaq, which still demand long time to decompress the files. Gzip still performs highly efficient to give 44% saving on BIN and 77% saving on HTML file, but better results are given by lzma for BMP and XML files, and consumed energy drops to 38% and 19% of the energy consumed for just simple receiving respectively. In the view of overall energy consumption, gzip offers the best results for all the files as showed in Fig 5.14.

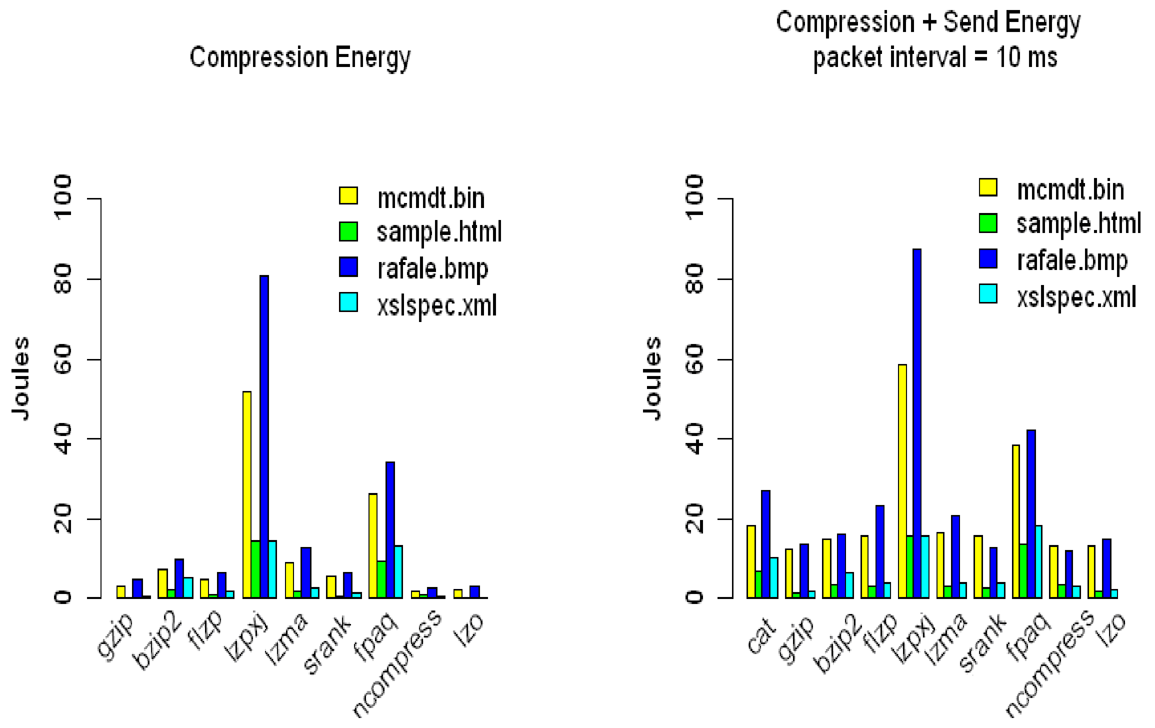

Fig 5.20 Energy required to send BIN, HTML, BMP and XML files

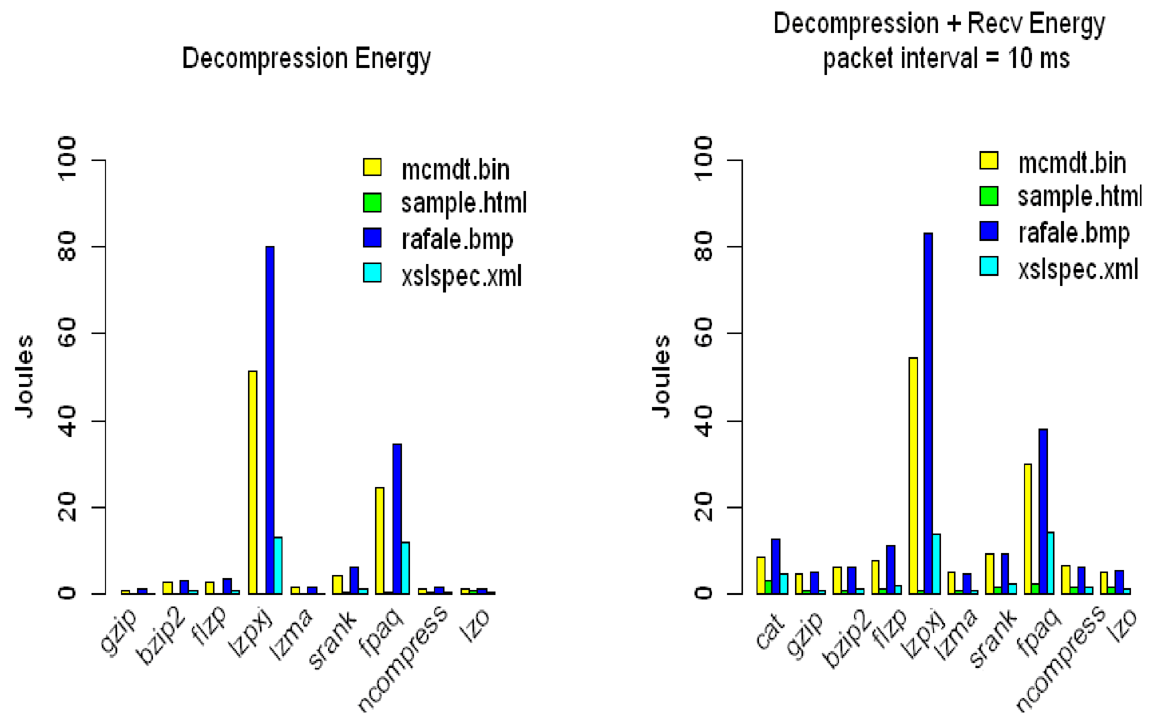

Fig 5.21 Energy required to receive BIN, HTML, BMP and XML files

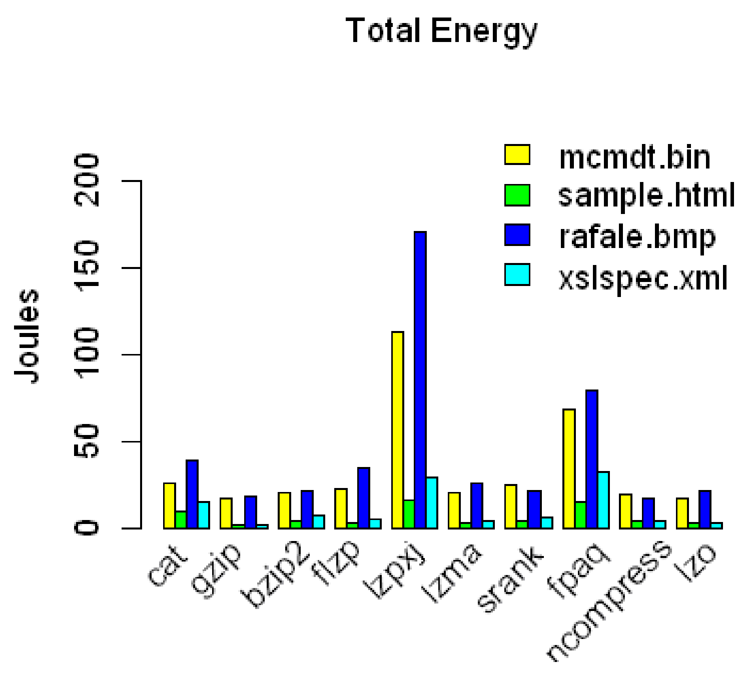

Fig 5.22 Total energy required to transmit BIN, HTML, BMP and XML files

5.3 Case Study of Web Sites

After the study on single file, examples are illustrated in Table 5.1 to assess the benefits in energy savings brought by using compression in a hand-held device. CNN, Facebook and MSN front pages captured on April 22nd 2009 are used as typical Internet traffic in our experiments. For all the pages, HTML files are concatenated with images, JavaScript, flash and gif files. The tests are carried on under a condition with 30ms packet sending interval to demonstrate a massive energy saving for typical usage on mobile device even under a relative slow link. In the table, the benefits of using compression to save energy when transmitting the web pages are presented with values for uplink, downlink and overall energy consumption. The consumed energy of direct copy and transmitting is showed by using cat to demonstrate a baseline of energy consumption. Compared to direct cat transmission, minimum energy consumed by certain compression programs is given in the following rows. The percentage of energy savings are also listed in bold type. For example, compression saves up to 50.36% energy when browsing MSN homepage. Even through energy saving degrees are highly depending on embedded contents of each web page, lzo, lzma and gzip are the best compression programs for energy savings, which generally offer the best energy savings for easy-compressed file transmission.

Table 5.3 Energy consumption on popular web pages

Pages	CNN	Facebook	MSN
Size	**862951B**	**609868B**	**633693B**
Uplink	**cat: 5.592 J**	**cat: 3.960 J**	**cat: 4.118 J**
	lzo: 3.205 J	**gzip: 1.523 J**	**lzo: 2.286 J**
	57.31%	38.46%	55.51%
Downlink	**cat: 2.611 J**	**cat: 1.853 J**	**cat: 1.928 J**
	lzma: 1.141 J	**gzip: 0.536 J**	**lzo: 0.971 J**
	43.70%	28.93%	50.36%
Overall	**cat: 8.203 J**	**cat: 5.813 J**	**cat: 6.046 J**
	gzip: 4.023 J	**gzip: 2.059 J**	**lzo: 3.257 J**
	49.04%	35.42%	53.87%

Chapter 6 CONCLUSIONS AND FUTURE WORK

One of the reasons to deploy compression in wireless communication for energy-aware purposes is that compression makes it possible to trade computation for communication energy savings. Compression is designed for reducing in file sizes by removing redundancy in the files. Indeed, the compression programs examined in our experiments achieve good performance on the files with BIN, HTML, BMP and XML extensions, however they always fail to compress such already highly compressed data as JPG, MP3, EXE and WMA files and even expand the file sizes. So, it indicates that compression schemes may cause significant energy losses instead of saving when blindly or carelessly chosen and applied. As demonstrated, energy consumed on uplink and downlink is asymmetric. We have also shown that, when carefully chosen and applied, some compression schemes allow an energy saving up to 57% and 50% on uplink and downlink respectively for visiting popular web sites on the N810 in IEEE802.11 WLAN environment, among which gzip, lzma and lzo are the generic compression programs providing great energy savings. Thus, it could be a tradeoff between sender and receiver that one of the ends uses more aggressive compute-intensive compression schemes to minimize the energy consumption on the other end.

As just mentioned, energy-aware communication should take the content of data into consideration, however it is not enough. Energy consumption on radio is highly depended on transmission time. As the quality of the radio link goes down, even small savings in file size can lead to substantial energy savings, since energy consumption per bit becomes increasingly significant. Through our experiments, compression is demonstrated as a great impact on energy saving in wireless networks when signal strength drops to certain degree.

In summary, a good data compression scheme for energy efficiency should be content-aware as well as aware of the link status to dynamically trade energy between computation and communication.

Overall, this study offers contributions in three aspects. First, we quantify the impacts of compression programs on popular Internet data in view of energy consumption. Second, the factors on reduction in energy in wireless networks and energy consumption tradeoffs associated with data compression and communication are presented. Based on that, we also address suggestions to how and when to compress for energy savings.

The study in this thesis is an investigation and evaluation of compression for energy-aware communication in IEEE802.11 networks. The results will be used for an energy efficiency-driven transmission that is capable of compressing and decompressing files on the fly. More precisely, the transmission could support feedback-driven compression which dynamically adjusts its parameters or select compression schemes based on circumstances. In order to achieve the goal, it would be necessary to understand more details as follows.

Regarding the study of compression, there are hundreds of compression programs available and some of them are designed for typical file types, such as xmill for XML pages. It would be useful to have such compression programs integrated into a content-aware transmission algorithm. Besides, file sizes definitely effect energy consumption of compression, so modeling the relationship is one of the interests.

This study focuses on evaluating the effects of compression on energy-aware communication. As we already know, energy consumption is related to transmission behaviors as well, so it is worth performing more experiments to investigate the energy consumption behavior of wireless interfaces not only in an ad hoc network but also in an infrastructure network for energy-aware design. Meanwhile, the study will pay more attention to different types of traffic such as broadcast and point-to-point data, not only limited to UDP packets. Moreover, energy consumption on 3G devices may significantly differ from the cases in our study, so it is interesting to perform similar experiments on 3G devices.

REFERENCES

Adobe Systems Incorporated. November 2006. “PDF Reference sixth edition” [Online]. Available at: http://www.adobe.com/devnet/acrobat/pdfs/pdf_reference_1-7.pdf (Referred 16.05.2009).

Adobe Systems Incorporated. November 2008. “SWF File Format Specification Version 10”. [Online]. Available at: http://www.adobe.com/devnet/swf/pdf/swf_file_format_spec_v10.pdf (Referred 16.05.2009).

Abrahams, J. Jun 1997. “Code and parse trees for lossless source encoding” Compression and Complexity of Sequences. pp 145-171.

Alistair Moffat & Andrew Turpin. 2002. “Compression and Coding Algorithms”. ISBN 978-0-7923-7668-2.

AMI Power Limited. “Compression History”. [Online]. Available at: http://www.ami.hk/image/products/DVR/History/Compression History.pdf (Referred 17.05.2009).

Barr. K. C. & Asanovic. K. 2006. “Energy-aware lossless data compression”. Syst. vol. 24, no. 3, pp. 250-291.

“bzip2”. 2009. [Online]. Available at: http://www.bzip.org (Referred 09.05.2009).

C Krintz and S Sucu. January 2006. “Adaptive on-the-fly compression,” IEEE Transactions on Parallel and Distributed Systems, vol. 17, pp. 15–24.

David Salomon, Giovanni Motta, David Bryant. 2007. “Data compression: the complete reference”. ISBN: 0387406972.

D.A. Huffman. 1952. “A method for the construction of minimum redundancy codes” Proc. IRE., 40(9):1098–1101.

D. Rand. June 1996. “The PPP Compression Control Protocol (CCP)”. IETF RFC 1962.

First International Workshop on Green Communications. Welcome note. [Online]. Available at: http://www.green-communications.net/icc09/home.htm

“gzip”. 2009. [Online]. Available at: http://www.gzip.org (Referred 09.05.2009).

ITU-T. 2006-09-29. “Data communication over the telephone network”. ITU. [Online]. Available at: http://www.itu.int/rec/T-REC-V/en (Referred 13.05.2009).

ITU-T. 2009-09-25. “Worldwide mobile cellular subscribers to reach 4 billion mark late 2008”. [Online]. Available at: http://www.itu.int/newsroom/press_releases/2008/29.html (Referred 13.05.2009).

Jacob Ziv, Abraham Lempel. May 1977. “A Universal Algorithm for Sequential Data Compression”. IEEE Transactions on Information Theory 23 (3). pp 337–343.

Jacob Ziv, Abraham Lempel. 1978. “Compression of individual sequences via variable-rate coding.” IEEE Transactions on Information Theory, IT-24(5). pp 530–536.

Jean-loup Gailly and Mark Adler. 2009. “gzip official pages”. [Online]. Available at: http://www.gzip.org/ algorithm.txt (Referred 09.05.2009).

Jukka Manner. 2006. “Jugi’s Traffic Generator(jtg)”. [Online]. Available at: http://www.cs.helsinki.fi/u/jmanner/software/jtg (Referred 09.05.2009).

J.-P. Ebert, B. Burns, and A. Wolisz. February 2002. “A trace-based approach for determining the energy consumption of a wlan network interface” in Proc. of European Wireless 2002, Florence, Italy, Feb. 2002, pp. 230–236.

Khalid Sayood. 2000. “Introduction to data compression”. The second edition.

“lzo”. 2009. [Online]. Available at: http://www.oberhumer.com/opensource/lzo (Referred 09.05.2009).

“lzma”.2009. [Online]. Available at: http://www.7-zip.org (Referred 09.05.2009).

Maddah Rakan, Sharafeddine Sanaa. August 2008. “Energy-Aware Adaptive Compression Scheme for Mobile-to-Mobile Communications” Spread Spectrum Techniques and Applications, 2008. ISSSTA '08. IEEE 10th International Symposium on , pp.688-691.

Markus Franz Xaver Johannes Oberhumer. 2008. “LZO -- a real-time data compression library”. [Online]. Available at: http://www.oberhumer.com/opensource/lzo/ (Referred 09.05.2009).

N810 specification. 2007. [Online]. Available at: http://europe.nokia.com/A4568578 (Referred 09.05.2009).

National Instruments. “Operating Instructions-CompactRIO cRIO-9215”. [Online]. Available at: http://digital.ni.com/public.nsf/fa8a679c1aabbd3486256d61004a378b/09a5914f6bc2a28348257172000af5b1/$FILE/9215_Operating_Instructions.pdf (Referred 11.05.2009).

Nemertes Research. “Internet Interrupted: Why Architectural Limitations Will Fracture the 'Net”. [Online]. Available at: http://www.nemertes.com/studies/internet_interrupted_why_architectural_limitations_will_fracture_net (Referred 09.05.2009).

P. Deutsch. May 1996. “DEFLATE Compressed Data Format Specification version”. IETF RFC 1951.

Peter Fenwick. 1996. “Symbol Ranking Text Compression”. [Online]. Available at: http://www.cs.auckland.ac.nz/~peter-f/FTPfiles/TechRep132.ps (Referred 05.05.2009).

Reijo Mäihäniemi, Efore Oyj. “ICT Getting Green”. Speech given to course Telecom-munications Forum in Helsinki university of technology. Oct 2008.

Robert G.Gallager, 1994. “Arithmetic coding”, supplementary notes. pp 1 [Online]. Available at: http://web.mit.edu/gallager/www/notes/notes3.pdf (Referred 06.05.2009).

R Xu, Z Li, C Wang, and P Ni. 2003. “Impact of data compression on energy consumption of wireless-networked handheld devices,” in IEEE International Conference on Distributed Computing Systems.

The final version of the Work Programme for ICT research in FP7 for 2009 and 2010. [Online]. Available at: http://cordis.europa.eu/fp7/ict/newsroom/library_en.html (Referred 09.05.2009).

APPENDIX A. RESULTS OF THE EXPERIMENTS

Packet sending interval for the following data is 10ms

Table A.4 Compression ratio, time and energy in compression and decompression for A10.jpg

Files	Ratio	Ratio/Time	C_Time(s)	D_Time(s)	C_Energy(J)	D_Energy(J)	Compress and Send Energy	Decompress and recv Energy	Total Energy
cat	1	-	-	-	-	-	18.252	17.529	35.781
gzip	1.0012-1.0018	0.2301-0.2631	3.808-4.352	3.406-4.049	1.488-1.722	0.716-0.930	19.261-19.464	17.745-17.959	37.022-37.309
bzip2	1.0011-1.0072	0.1111-0.1511	6.626-9.503	4.385-5.360	4.199-5.467	2.017-2.248	21.951-23.137	19.047-19.192	40.998-42.329
flzp	0.9436	0.1813	5.206	4.100	2.677	1.515	21.512	19.583	41.094
lzpxj	1.0089	0.0253	39.858	41.738	28.946	30.488	46.562	47.386	93.948
lzma	0.9924-0.9963	0.0868-0.1207	8.245-11.471	11.743-16.698	4.780-7.011	1.327-1.506	22.638-24.859	18.444-18.677	41.200-43.264
srank	0.7206-0.9960	0.1105-0.1640	6.049-6.521	3.897-4.892	2.581-2.728	1.647-2.055	20.425-27.374	19.083-25.306	39.507-53.518
fpaq	0.9914	0.0831	11.927	11.505	8.351	8.041	26.279	25.239	51.518
ncompress	0.8145	0.1343	6.064	4.295	1.823	1.135	28.765	25.694	54.459
lzo	0.9999-0.9999	0.1401-0.3281	2.698-7.139	3.026-4.451	0.706-3.348	0.607-1.737	18.480-21.122	17.692-18.787	36.188-39.986

Table A.5 Compression ratio, time and energy in compression and decompression for mean.wma

Files	Ratio	Ratio/Time	C_Time(s)	D_Time(s)	C_Energy(J)	D_Energy(J)	Compress and Send Energy	Decompress and recv Energy	Total Energy
cat	1	-	-	-	-	-	31.166	29.910	61.075
gzip	1.0318-1.0330	0.1229-0.1420	7.961-8.390	6.658-12.014	3.264-3.633	1.800-2.300	33.166-33.551	26.537-30.950	59.537-64.172
bzip2	1.0188-1.0289	0.0691-0.0855	11.968-14.889	7.532-8.622	7.318-8.672	3.020-4.190	37.462-38.661	32.778-32.965	70.293-71.457
flzp	0.9639	0.1172	8.227	8.165	5.182	3.020	37.191	33.726	70.917
lzpxj	1.0451	0.0151	69.006	72.071	51.234	52.240	80.831	80.599	161.390
lzma	1.0193-1.0305	0.0507-0.0691	14.888-20.306	7.569-8.896	8.681-12.550	2.594-3.854	38.690-42.492	31.316-32.889	70.413-73.835
srank	0.7550-1.0304	0.0385-0.0796	12.580-19.543	9.471-11.095	5.554-7.549	4.251-4.723	35.496-48.605	32.974-44.108	68.470-92.714
fpaq	1.0232	0.0464	22.058	20.190	15.232	14.654	45.386	43.580	88.966
ncompress	0.8717	0.0779	11.190	7.641	3.322	2.232	48.167	42.639	90.806
lzo	1.0014-1.0047	0.0758-0.1579	6.650-13.259	6.316-7.062	2.048-18.173	1.531-2.731	32.857-37.684	31.080-32.231	63.987-69.273

Table A.6 Compression ratio, time and energy in compression and decompression for heart.mp3

Files	Ratio	Ratio/Time	C_Time(s)	D_Time(s)	C_Energy(J)	D_Energy(J)	Compress and Send Energy	Decompress and recv Energy	Total Energy
cat	1	-	-	-	-	-	49.274	47.287	96.561
gzip	1.0091-1.0115	0.0810-0.0912	11.074-12.493	9.267-10.442	5.128-5.588	2.462-3.248	53.478-53.852	48.801-49.497	102.322-103.499
bzip2	1.0079-1.0163	0.0441-0.0490	20.575-23.160	11.124-13.962	12.645-13.233	5.832-6.427	60.849-61.255	52.155-52.614	133.398-113.584
flzp	0.9492	0.0773	12.287	11.244	7.720	4.401	59.154	53.732	112.886
lzpxj	1.0161	0.0095	107.247	110.133	80.248	79.320	128.289	125.405	253.94
lzma	1.0040-1.0151	0.0297-0.0426	23.623-34.151	10.192-11.513	13.3434	3.864	61.802	49.997	112.729
srank	0.7296-1.0151	0.0325-0.0487	17.904-22.479	14.541-19.034	8.022-9.397	6.279-7.346	56.757-76.300	52.412-71.427	109.169-147.538
fpaq	0.9994	0.0315	31.752	31.185	23.388	22.748	72.233	69.604	141.837
ncompress	0.8345	0.0472	17.673	11.268	5.279	3.124	78.698	68.753	147.451
lzo	1.0002-1.0040	0.0501-0.1158	8.824-20.028	8.267-9.212	2.737-10.829	2.202-2.569	51.544-60.022	48.920-49.261	100.634-108.699

Table A.7 Compression ratio, time and energy in compression and decompression for qq.exe

Files	Ratio	Ratio/Time	C_Time(s)	D_Time(s)	C_Energy(J)	D_Energy(J)	Compress and Send Energy	Decompress and recv Energy	Total Energy
cat	1	-	-	-	-	-	106.040	101.756	207.795
gzip	1.0045-1.0046	0.0304-0.0393	25.569-28.674	20.963-21.530	11.357-13.836	5.810-11.87	116.104-118.575	106.289-112.342	222.484-230.917
bzip2	0.9960-0.9988	0.0195-0.0221	45.524-51.279	27.051-31.581	27.967-30.173	13.964-14.524	133.426-135.575	115.299-115.992	248.898-250.979
flzp	0.9433	0.0365	27.349	25.620	17.776	10.164	129.315	117.161	246.477
lzpxj	0.9960	0.0039	242.329	243.223	179.627	177.052	285.263	278.387	563.650
lzma	0.9919-0.9922	0.0133-0.0195	51.110-74.749	22.123-25.266	29.168-47.080	9.028-9.389	135.242-153.235	110.747-111.107	246.051-264.343
srank	0.7135-0.7136	0.0110-0.0154	54.637-64.476	44.569-49.594	18.324-24.227	17.407-18.976	165.75-171.661	158.842-160.418	325.471-331.867
fpaq	0.9973	0.0091	78.258	67.407	54.293	50.334	159.783	151.529	311.312
ncompress	0.8147	0.0244	40.868	25.320	10.834	7.559	174.492	152.828	327.32
lzo	1.0032-1.0042	0.0232-0.0502	21.076-43.375	19.810-20.896	6.545-25.535	5.527-6.493	111.421-130.300	106.110-106.490	217.965-236.731

Table A.8 Compression ratio, time and energy in compression and decompression for nb.swf

Files	Ratio	Ratio/Time	C_Time(s)	D_Time(s)	C_Energy(J)	D_Energy(J)	Compress and Send Energy	Decompress and recv Energy	Total Energy
cat	1	-	-	-	-	-	81.829	78.513	160.342
gzip	1.1085-1.1160	0.0573-0.0626	17.795-19.469	9.141-9.619	7.971-9.9952	2.280-4.941	81.327-82.958	72.324-74.957	153.652-156.350
bzip2	1.1148-1.1804	0.0349-0.0374	29.782-33.687	14.833-19.041	18.851-19.546	7.475-8.262	88.314-91.985	74.129-77.578	162.443-169.563
flzp	1.1350	0.0678	16.739	12.093	11.030	5.282	82.808	74.137	156.945
lzpxj	1.2434	0.0081	154.317	153.074	115.705	113.324	181.224	176.175	357.399
lzma	1.2058-1.3313	0.0219-0.0367	32.842-60.699	10.073-12.028	19.379-40.534	3.999-4.250	84.521-101.725	62.699-68.923	148.940-164.424
srank	0.8150-0.9054	0.0172-0.0245	36.957-47.288	27.265-35.379	14.985-17.416	9.879-11.155	104.962-117.373	97.468-106.531	202.430-223.659
fpaq	1.0410	0.0167	62.213	50.392	42.702	37.220	120.959	112.291	202.430
ncompress	0.8676	0.0355	24.444	13.060	6.342	3.339	70.752	93.412	164.164

lzo	1.0743-1.1480	0.0373-0.0810	13.894-29.935	7.611-8.791	5.019-17.339	1.881-2.406	77.535-89.713	70.259-74.851	149.140-161.835

Table A.9 Compression ratio, time and energy in compression and decompression for sample.html

Files	Ratio	Ratio/Time	C_Time(s)	D_Time(s)	C_Energy(J)	D_Energy(J)	Compress and Send Energy	Decompress and recv Energy	Total Energy
cat	1	-	-	-	-	-	6.858	3.202	10.060
gzip	4.041-4.806	2.585-4.946	0.919-2.486	0.834-1.385	0.101-1.119	0.117-0.360	1.794-2.547	0.387-1.975	1.432-4.054
bzip2	4.930-5.466	1.116-1.429	3.593-4.672	1.399-2.365	2.084-4.672	0.502-0.734	3.475-4.083	2.631-3.970	5.106-7.833
flzp	3.420	1.580	2.164	1.647	0.925	0.543	2.930	1.197	4.127
lzpxj	5.708	0.293	19.459	17.208	14.533	12.230	15.755	14.012	29.767
lzma	2.6865-2.8808	0.243-1.560	3.363-24.734	1.073-1.520	1.827-18.914	0.091-0.290	3.315-20.054	2.673-16.235	4.614-32.984
srank	1.6701-1.8703	1.077-1.646	1.942-2.310	2.595-3.044	0.757-0.626	0.667-0.842	2.771-3.515	1.340-2.313	3.190-5.548
fpaq	1.4792	0.131	9.386	9.259	9.386	6.140	13.410	11.012	24.422
ncompress	1.5848	1.347	0.836	1.882	0.836	0.331	3.442	2.069	5.511
lzo	2.0464-2.4843	1.006-4.227	0.109-2.477	0.802-1..178	0.109-2.477	0.165-0.183	2.172-4.106	1.919-3.050	3.814-6.516

Table A.10 Compression ratio, time and energy in compression and decompression for flash.dpf

Files	Ratio	Ratio/Time	C_Time(s)	D_Time(s)	C_Energy(J)	D_Energy(J)	Compress and Send Energy	Decompress and recv Energy	Total Energy
cat	1	-	-	-	-	-	96.139	92.251	188.390
gzip	1.1598-1.1810	0.0521-0.0596	20.090-22.685	11.837-16.060	8.880-11.470	4.211-7.253	90.284-92.335	81.793-84.826	172.224-176.232
bzip2	1.1768-1.1891	0.0293-0.0334	35.207-40.588	19.664-23.484	21.656-23.871	9.984-10.709	102.251-104.265	87.695-88.200	190.102-191.021
flzp	1.0786	0.0457	23.598	18.610	13.306	7.605	101.851	92.544	194.395
lzpxj	1.2115	0.0066	183.468	187.417	137.736	135.606	216.565	211.225	427.790
lzma	1.1994-1.2213	0.0145-0.0301	39.816-83.984	16.505-17.787	22.928-57.596	6.346-6.859	102.555-135.791	81.367-83.244	185.799-217.522
srank	0.8502-1.2213	0.0300-0.0428	27.616-28.796	21.805-23.053	12.926-13.682	9.523-10.932	91.479-125.246	85.792-117.318	177.271-242.516
fpaq	1.118	0.0191	58.600	58.012	43.311	42.972	128.736	124.918	253.654
ncompress	0.834	0.0282	29.798	20.810	6.478	5.889	146.002	133.307	279.309
lzo	1.118-1.175	0.0312-0.0698	16.809-37.416	13.546-14.730	5.042-22.447	3.688-5.382	89.280-103.793	82.095-85.872	173.132-186.208

Table A.11 Compression ratio, time and energy in compression and decompression for rafale.bmp

Files	Ratio	Ratio/Time	C_Time(s)	D_Time(s)	C_Energy(J)	D_Energy(J)	Compress and Send Energy	Decompress and recv Energy	Total Energy
cat	1	-	-	-	-	-	87.756	84.192	171.948
gzip	2.6681-3.3080	0.1373-0.3796	8.208-24.100	5.203-3.375	3.992-16.124	1.139-2.425	32.935-42.586	26.596-32.611	60.985-70.395
bzip2	4.2972-4.6614	0.2505-0.3283	13.088-18.612	8.220-9.678	9.712-12.434	3.157-3.621	29.944-31.280	21.471-22.698	52.267-53.284
flzp	1.5905	0.1172	13.757	6.891	6.366	3.501	61.404	56.298	117.702
lzpxj	4.0631	0.0373	109.008	108.216	80.903	79.894	102.447	100.561	203.008
lzma	3.4509-4.2572	0.0410-0.1719	20.078-103.694	5.027-7.190	12.782-75.870	1.874-2.202	38.148-96.440	21.599-26.535	64.683-118.163
srank	2.2069-4.5272	0.1109-0.3487	14.466-20.477	21.373-29.781	6.546-9.502	6.341-8.414	27.199-48.065	26.066-44.92	53.265-92.993
fpaq	3.4037	0.0723	47.060	46.878	34.372	34.108	59.826	59.043	118.869
ncompress	2.8718	0.3053	9.406	7.340	2.650	1.800	41.074	37.238	78.313
lzo	1.7122-2.8707	0.056-0.3071	7.047-48.147	3.790-4.505	2.451-33.492	0881-1.654	41.172-65.606	30.370-49.948	80.084-103.924

Table A.12 Compression ratio, time and energy in compression and decompression for xlspec.xml

Files	Ratio	Ratio/Time	C_Time(s)	D_Time(s)	C_Energy(J)	D_Energy(J)	Compress and Send Energy	Decompress and recv Energy	Total Energy
cat	1	-	-	-	-	-	33.479	32.119	65.598
gzip	5.5916-7.2995	2.2881-4.4450	1.365-3.190	0.903-1.424	0.344-1.682	0.103-0.496	5.220-6.317	4.492-5.832	10.023-12.149
bzip2	7.6403-9.8951	0.8627-1.0881	7.022-11.470	2.820-3.270	5.008-8.195	0.856-1.039	9.383-11.573	4.188-5.052	14.600-15.857
flzp	4.8331	1.5831	3.053	2.048	1.695	0.933	8.611	7.568	16.178
lzpxj	10.2946	0.4919	20.929	18.607	14.523	13.254	17.770	16.368	34.139
lzma	7.4499-9.8221	0.2433-1.8569	4.012-40.374	0.762-1.699	2.719-29.897	0.140-0.362	7.206-33.300	3.420-4.583	11.547-36.871
srank	2.8366-9.7774	0.4631-3.5986	2.717-6.125	4.118-7.623	1.286-2.683	1.337-2.424	4.773-14.467	4.652-13.661	9.424-27.961
fpaq	2.1003	0.1066	19.694	17.058	13.339	12.148	29.254	27.415	56.669
ncompress	4.1818	1.5130	2.764	2.110	0.772	0.377	11.1	9.827	20.927
lzo	3.8141-6.3185	0.9881-5.4829	1.235-6.353	0.603-1.304	0.135-3.302	0.229-0.799	6.409-9.116	5.774-8.780	12.669-17.896

Table A.13 Compression ratio, time and energy in compression and decompression for mcmd.bin

Files	Ratio	Ratio/Time	C_Time(s)	D_Time(s)	C_Energy(J)	D_Energy(J)	Compress and Send Energy	Decompress and recv Energy	Total Energy
cat	1	-	-	-	-	-	58.382	56.010	114.392
gzip	1.9751-2.1208	0.1074-0.2903	6.803-19.753	2.620-4.229	3.212-13.507	0.774-2.268	31.920-40.970	27.208-29.079	59.829-69.583
bzip2	2.2186-2.4854	0.1812-0.2013	11.206-13.442	6.632-7.843	7.410-8.220	2.781-3.127	31.248-34.032	25.397-27.964	56.651-61.996
flzp	1.6626	0.2057	8.084	5.632	4.757	2.678	39.788	36.283	76.071
lzpxj	2.7586	0.0392	70.339	70.051	51.946	51.226	73.059	71.479	144.538
lzma	2.4807-3.0625	0.0545-0.1655	15.121-56.143	3.881-6.083	9.140-40.629	1.345-1.727	31.512-59.657	19.646-24.090	53.182-79.358
srank	1.5882-1.7502	0.0813-0.1633	10.653-19.540	13.827-19.656	5.593-7.705	4.435-5.406	39.066-44.376	36.545-40.584	75.611-84.960
fpaq	1.5104	0.0404	37.364	33.739	26.368	24.467	64.928	61.457	126.385
ncompress	1.5686	0.1644	9.539	5.554	1.759	1.307	46.945	41.615	88.56
lzo	1.5087-2.0591	0.0729-0.282	6.301-27.776	2.856-3.652	1.755-18.351	0.550-1.347	36.342-47.017	30.721-37.856	69.041-79.294