Energy Consumption Trade-offs for XML Compression on Embedded Devices

Jiva N. Bagale, John P. T. Moore, Antonio D. Kheirkhahzadeh, Yasmine Rosunally
School of Computing and Technology
University of West London
St. Mary's Road, London W5 5RF
{bagajiv, moorejo, kheiant, rosuyas}@uwl.ac.uk

***Abstract*—Wireless data transmission over sensor networks is known to consume a significant share of energy compared to data computation on the device itself. Thus data compression techniques have been used to reduce the data size at the cost of extra processing time on the device. This paper investigates the energy consumption trade-off of XML compression and data communication on embedded devices with low-bandwidth network.**

I. Introduction

The evolution of sensor technologies has allowed embedded devices to collect a range of information about a user's location, activity or their environment which will then be communicated over the network. Data compression can be used to reduce the number of bits communicated over network and to minimise overall energy consumption. We aim to investigate this further by analysing the impact of compression on the overall energy consumption. Extensible Markup Language (XML) is a flexible data format and is widely used as a data exchange and storage format on the web and sensor networks [1]. It is very verbose because of the repetitive usage of tags to represent data structure. As a result of this, alternative formats such as JavaScript Object Notation (JSON) are considered for their conciseness. However XML is still used by many new and old applications and XML Schema (metadata) can be used to define sensor data types and restrictions. The extra knowledge about data types can also be used to efficiently compress XML data to make it competitive in terms of size. Schema-aware compression is very efficient and the compression ratio for XML data using well defined schema is more than 80% [2]. If it requires more energy to compress the data than to communicate the uncompressed data the compression process will not be beneficial.

The energy cost of wireless communication is considered to be significantly higher than on device computation. It requires over 1000 times more energy to transmit a single bit of data than a single 32-bit computation [3]. Thus, it may be efficient in terms of energy to perform 1000 computations in order to compress data by 1 bit. Existing researches have found that compression techniques are mostly beneficial for saving network bandwidth and energy consumption. However the degree of benefits depends upon the compression ratio and speed. In [4], *gzip* compression software (based on LZ77) was found to be superior to *bzip2* (based on BWT) and *compress* (based on LZW) in terms of energy saving in a hand-held device using a variety of file types. The algorithm with the best compression ratio was not necessarily the most efficient in terms of energy savings despite having a smaller download time (and less energy spent during download) due to the increased energy cost in decompression. Similarly authors in [5] compared various XML compression techniques including gzip, XMILL and PAQ to Tinypack. The experiments found that the total time for sending compressed data is always lower than sending uncompressed data. PAQ reduces the data size most but it uses huge amount of memory and spends more time to process than to send uncompressed data

Earlier results imply that if it requires too much time or memory to compress data, it may not be beneficial overall in terms of energy despite significantly reducing the data. Thus compression must be performed to balance the trade-off between compression ratio, compression speed and CPU and memory consumption in order to provide overall energy consumption savings. Earlier work has considered schema-aware XML compression to achieve compression ratio better than text-based XML compression using simple and efficient compression techniques [2, 6]. In this paper, we will investigate this further and evaluate the energy consumption of schema-aware compression techniques to justify its advantages over text-based compression techniques or when there is no compression at all.

II. Methodology

The embedded device used for the experiment is the Raspberry Pi single board computer (SBC) which is running Raspbian OS and has 700MHz ARM processor and 512MB RAM. Energy consumption was measured using an ODROID smart power device which has output of 3-5.25V DC and can measure current or watts consumed by the connected devices. In these experiments, we compare XML compression techniques Packedobjects (PO), Efficient XML Interchange (EXI), zlib and Libxml2. Tinypack was considered for comparison as well but its implementation and the test XML data-sets weren't publicly available. We evaluated WBXML, XMILL, XMLPPM and DTDPPM along with PO and EXI in an earlier experiment to compare their compression speed and ratio. They were found to have slower compression speed and/or worse compression ratio than PO & EXI and thus aren't

978-3-901882-70-8

TABLE I
TEST XML FILES, DATA TYPES AND COMPRESSION RATIO

Name	Size(Byte)	Data Types	po	exi	zlib
ping	72	enumerated	72	24	1.09
timestamps	246	unix-time	16.4	2.51	1.7
router-addnet	510	string	5.48	5.31	2.16
sensor	549	numeric	21.96	17.16	2.53
iptel-ethinfo	627	numeric	17.42	16.08	2.2
iptel-devinfo	639	string	7.26	6.09	1.97
switch-config	678	string & ipv4-address	8.17	7.98	2.65
purchaseorder	738	string & numeric	7.38	6.31	2.31
router-disc	767	string & numeric	8.16	9.13	2.46
router-qos	817	string	6.92	5.79	3.13
personnel	856	string	11.41	9.41	2.89
menu	860	enumerated & string	78.18	47.78	2.88
unsafenumbers	1,155	numeric	15.61	16.04	5.04
temper-sens	1,307	string & numeric	16.97	18.41	3.72
vehicles	5,142	string & numeric	13.64	14.4	4.55

considered for this experiment. Libxml2 is used to serialise XML without compression to make it suitable for network transfer. PO [2] provides schema-aware XML compression by defining domain-specific subset XML Schema. It focuses on compressing data for network transfer so it is fast, light-weight and highly portable. It packs data up to bit level allowing efficient compression especially for smaller data sizes. Common data types such as string and integer are supported along with domain-specific ones such as hex-string, IPv4-address and Unix-time and further types can be added. EXI also provides schema-aware XML compression and is recommended by the World Wide Web Consortium (W3C). It can use data type information and other restrictions defined in the Schema to reduce the XML file size. It can utilise XML Schema, RELAX NG schema or DTD to optimise compression but we will compare with EXIProcessor, a schema-informed Java implementation of EXI. Although it provides compression ratio similar to PO as it also performs bit-level compression, the compression speed is slower and also require large memory. Zlib provides compression based on DEFLATE algorithm which itself is a variation of LZ77 algorithm. It is highly portable compression tool and the memory consumption is independent of input data size.

Fig. 1. Odroid smart power measurement tool

The XML corpus consists of a variety of data types including string, integer, decimal, numeric-string, IPv4 address, UNIX timestamps and enumeration as shown in table I. All XML files also have a XML Schema which will be used by PO and EXI during compression. The file size ranges from 72 bytes to 5142 bytes and larger files were not considered as embedded devices on sensor networks won't normally communicate those sizes. In order to measure energy consumption of data compression we combine data from two sources: first we measure the time expended using the GNU/Linux command line tool *time* and second we measure the average energy consumed by using the ODROID smart power device as shown in figure 1. In order to measure energy consumption of network transfer, we first measure energy consumption of network transfer per second and then calculate the energy consumption for the actual network transfer based on the total time taken 1 and energy consumed per second. We measured energy consumed when transferring various compressed and uncompressed files with the network utility program *ttcp* and then calculate the average energy consumed by network transfer in a one second period. The data transfer time for communicating compressed and uncompressed XML files is then calculated using a theoretical network speed. The energy consumption measured for both compression and network transfer activity includes the energy consumed by respective devices when idle.

$$time(s) = datasize(bits)/networkspeed(bit/s) \quad (1)$$

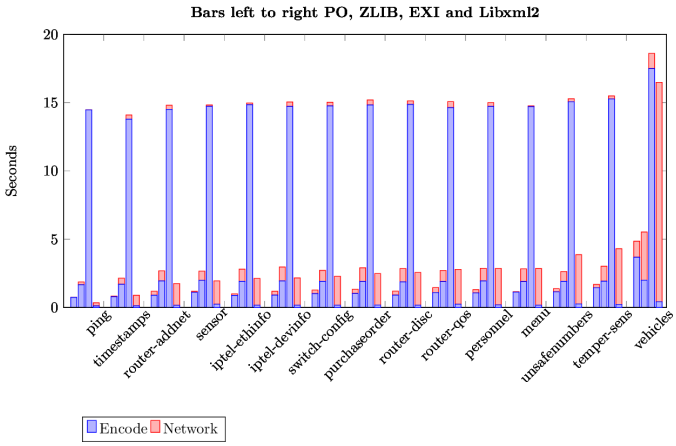

Fig. 2. Overall data compression and network transfer time

III. RESULTS

In this section, we examine experimental results to examine the compression ratio, compression time and energy consumption and evaluate if the total energy consumption after network transfer can be reduced by compression. We assume that all the files that are compressed on the embedded devices are uncompressed on a server later so we only focus on the compression and network transfer of the data. We are assuming that packet size and data rate are fixed and thus energy cost of data transfer over network transfer is linearly proportional to the size of the data. Thus the energy savings in the data transfer

from reduced data sizes must be greater than the extra energy spent on data compression on the device.

In table I above, we show the compression ratio for PO, EXI and zlib for different XML files listed in ascending order of size in bytes. Compression ratio is presented as the ratio of uncompressed data size to compressed size and thus the higher ratio means the better compression and vice verse. PO has the best compression ratio and EXI has similar ratio but ZLIB has poor ratio. EXI has a compression ratio similar to PO for smaller sized files and slightly better than PO for larger files. Zlib has the worst compression ratio of the three techniques and it compresses string data types better than non-string types. The data transfer speed used in the experiment is 250kbps to reflect the bandwidth of low-rate IEEE 802.15.4 standard such as 6LoWPAN used by embedded and sensor networks. The transfer time is calculated theoretically as discussed above in methodology. Figure 2 shows the overall time spent for compressing the files on a Raspberry Pi and transferring the data over a 250kbps network for three compression techniques along with the uncompressed data transfer. It takes a negligible amount of time to serialise uncompressed data as it is only XML to string conversion. The network transfer time for the uncompressed data is the biggest compared to the transfer time for any of the three compressed formats. However, zlib and EXI still spend more time on the overall to send compressed data with compare to uncompressed data. Although zlib compresses considerably faster than EXI, it still can't save time overall compared to uncompressed data due to its poor compression ratio. EXI is very ineffective when used on embedded devices despite having an excellent compression ratio because of the extremely slow compression speed. PO however has the best compression ratio and compresses significantly faster as well. Thus, PO is the best technique which provides time savings overall in terms of compression time and data transfer. The only exception however where PO compression can't provide time savings is for the smallest file which is 72 bytes. This is because of the start up time of PO being bigger than the actual data transfer time of an uncompressed file. Although the time savings for smaller XML files is not huge, it is significant for larger files.

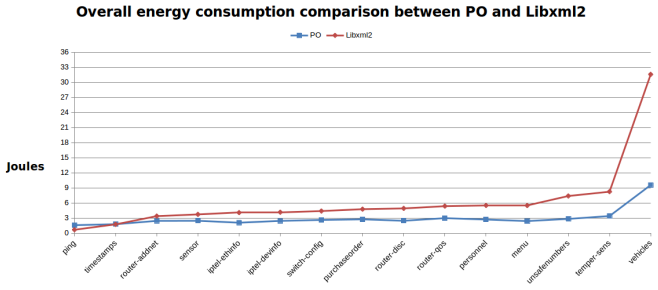

Fig. 3. Overall energy comparison for compressed and uncompressed data

The average energy consumption per second for idle state and compression processes is shown in table II. The energy consumption for network transfer is assumed to be 1.915

Idle	Libxml2	EXI	Zlib	PO
1.75	1.86	1.96	2.04	2.06

TABLE II
AVERAGE ENERGY CONSUMPTION PER SECOND DURING COMPRESSION IN WATT

including the idle consumption which represents energy consumption of typical low-power wireless modules currently on the market (e.g., Crossbow TelosB) [7]. Both EXI and zlib spend more time on the overall than when uncompressed data is communicated over the network and energy consumption increases linearly with time. Thus the overall energy consumption of only PO compression and Libxml2 for uncompressed data is shown in figure 3. PO consumes more energy during compression and less during the data transfer. Libxml2 consumes less energy during data serialisation and more energy during data transfer. On the overall PO consumes less energy with compare to uncompressed data serialisation and network transfer with the exception of the smallest XML file. The energy savings increases gradually with the increase of file size and the saving is significantly more in the case of the largest file used. PO is saving energy because it consists of the best compression ratio and also the fastest compression speed. EXI fails to save energy because of its slow compression speed and zlib fails to save energy because of its poor compression ratio and slow compression speed.

IV. CONCLUSION

EXI and zlib are found to be not suited for embedded devices as they spend too much energy during compression which negates the energy savings from the data size reduction. PO provides overall energy savings compared to uncompressed data transfer over a 250Kbps network. EXI doesn't provide energy savings on the overall despite having excellent compression ratio. Thus data compression can save energy only when the compression ratio is good and the compression speed is also fast.

REFERENCES

[1] W3C, "Extensible Markup Language (XML)." http://www.w3.org/XML, 1996.

[2] J. Moore, A. Kheirkhahzadeh, and J. Bagale, "Towards Markup-Aware Text Compression," in *Data Compression Conference (DCC), 2014*, pp. 417–417, March 2014.

[3] K. C. Barr and K. Asanovic, "Energy-aware Lossless Data Compression," *ACM Trans. Comput. Syst.*, vol. 24, pp. 250–291, Aug. 2006.

[4] R. Xu, Z. Li, C. Wang, and P. Ni, "Impact of data compression on energy consumption of wireless-networked handheld devices," in *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on*, pp. 302–311, May 2003.

[5] T. Szalapski, S. Madria, and M. Linderman, "TinyPack XML: Real time XML compression for wireless sensor networks," in *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, pp. 3165–3170, April 2012.

[6] A. Kheirkhahzadeh, J. Moore, and J. Bagale, "XML-compression techniques for efficient network management," in *Globecom Workshops (GC Wkshps), 2013 IEEE*, pp. 996–1000, Dec 2013.

[7] A. Ayadi, P. Maille, and D. Ros, "TCP over Low-Power and Lossy Networks: Tuning the Segment Size to Minimize Energy Consumption," in *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*, pp. 1–5, Feb 2011.