

# Poster Abstract: Compressing Software for Energy-Efficient Reprogramming of Wireless Sensor Networks

Nicolas Tsiftes  
Swedish Institute of Computer Science  
nvt@sics.se

## Abstract

Distribution of software over-the-air in sensor networks requires radio transmission energy that is proportional to the size of the software image. Because the radio transceiver is one of the most energy consuming devices on sensor nodes, it is important to reduce the size of the software. This poster presents the results of using a variety of different data compression algorithms to trade-off transmission energy with the computation energy that is consumed when decompressing the software. I show that by using data compression, we can save a considerable amount of energy when reprogramming sensor networks.

## 1 Introduction

Reprogramming wireless sensor networks requires that software is reliably distributed to the recipients and that the method does not waste energy. As the need to do reprogramming over-the-air emerged, software images consisting of both the operating system and applications were initially distributed regardless of the scale of the change. In order to make the reprogramming more energy-efficient and fast, methods based on virtual machines [6], dynamically linkable modules [2], and version differences [4, 5] have been developed.

Virtual machine languages can be adapted to different types of applications to reduce the instruction set size. However, the interpretation and execution of VM codes cause a significant run-time overhead. In contrast, version differences and dynamically linkable modules are used to reprogram sensor nodes with native code that can be executed directly by the processor. Version differences are effective when the changes are small and the network is homogeneous. The administrative burden will increase if the network is heterogeneous or if different versions exist. The Contiki operating system uses run-time dynamic linking of native software to enable separate distribution of new or updated applications. This approach also has the advantage that the no system restart is required upon installment.

By using data compression we can reduce the radio transmission energy by up to 50% on the software used for the experiments. This reduction is made at the expense of decompressing the files at the sensor nodes. In this paper, I show that this trade-off is beneficial in terms of energy-efficiency on both the Scatterweb ESB and the Tmote Sky board.

## 2 Data Compression in Sensor Networks

General-purpose compression software typically relies on memory on the order of hundreds of kilobytes for statistical models, sliding-windows, I/O buffers, or transformation buffers. The memory-constrained environments of sensor nodes require that special consideration must be taken regarding data structures and transformation buffers in the implementations. To measure and compare the energy-efficiency with different algorithms, I have implemented a selection of decompression algorithms in Contiki. The set of implementations consists of AC, GZIP, LZARI, LZOIDX, SBZIP, and VCDIFF. AC uses adaptive arithmetic coding with 32-bit integers that are scaled when the intervals converge to certain ranges. The GZIP decompressor uses the DEFLATE algorithm [1], which is a combination of Huffman Coding and Ziv-Lempel sliding-window compression. The SBZIP software that I have designed for sensor nodes is a three step algorithm that is composed of the Burrows-Wheeler Transform on 512 byte blocks, Move-To-Front Coding, and Huffman coding with a constant statistical model. VCDIFF is a decoder for the generic difference compression format of the same name. It is capable of encoding files using sliding-window compression. In addition I have ported the existing implementations of LZARI and LZOIDX to Contiki.

## 3 Evaluation

In this preliminary evaluation I provide the results of using data compression on the Scatterweb ESB and the Tmote Sky sensor boards. Both boards have microcontrollers that are based on the MSP430 architecture. The compression methods have been evaluated in terms of energy consumption, execution time, compression ratio, and memory footprint. The software set used in the experiments consists of six modules from the Contiki CVS repository that represent both device drivers and applications.

### 3.1 Energy-Efficiency

Figure 3.1 shows the energy costs of radio transmission with the TR1001 transceiver, and the decompression energy on the MSP430F1611 microcontroller with 3.5V input voltage. The average Deluge protocol overhead [3] is assumed for the radio trans-

mission calculations.

The measurements show that all of the implementations reduce the energy consumption. Most energy is saved by using GZIP due to its low compression ratio and relatively fast execution.

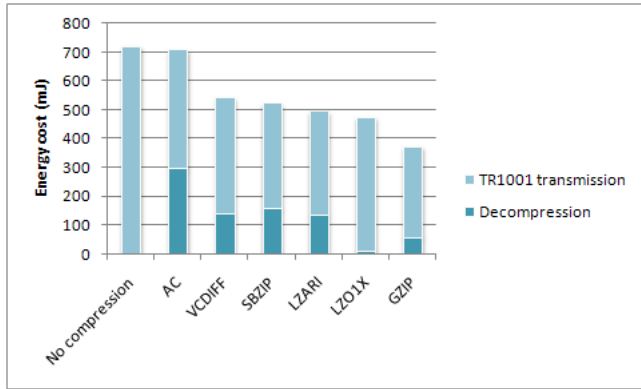


Figure 1. Energy-efficiency of the compression techniques compared to using no compression.

### 3.2 Memory Footprints

Sensor boards are typically constrained with memory ranging from 1 to 10 kilobytes, and flash memory that is an order of magnitude higher. The memory space is also shared with the Contiki kernel and other applications. For this reason, it is important that the decompression software is optimized to use significantly less memory than what is available on the boards.

Table I. Memory and storage requirements (in bytes) of the implementations.

Program	Memory footprint	Software size
AC	552	1424
VCDIFF	1623	2890
SBZIP	1645	4358
LZO1X	2211	5146
GZIP	2357	7696
LZARI	2983	2328

Table I shows the memory footprints and the flash memory required to store the software. Only three of the implementations are suitable for the ESB that has 2 kilobytes of RAM. The most memory-consuming implementation uses roughly 30% of the 10 kilobytes of memory that are available in the Tmote Sky.

### 3.3 Decompression Times

The decompression times, as shown in Figure 3.3, are quite long for all implementations except LZO1X, which has been optimized for speed. The performance of the GZIP decompressor is the second highest because bytes are often copied from the sliding window instead of being decoded with the Huffman algorithm. The arithmetic decoder has a trade-off between memory requirements and computational complexity. A larger data structure to store and update the statistical model would improve the speed significantly, and thus make it more energy-efficient.

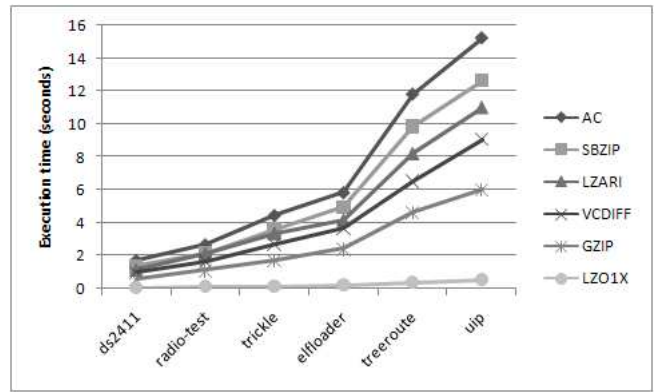


Figure 2. Decompression times on the Tmote Sky with 3.5V input.

## 4 Conclusion

Our experiments show that the GZIP decompressor is the most energy-efficient of the implementations when used with a TR1001 radio transceiver, and saves on average 47%. The difference in energy savings when compared to the Tmote Sky is significant. Although LZO1X has the highest compression ratio, it is the most energy-efficient when using the CC2420 radio transceiver. The reason for this is the low energy cost per byte for transmission, and the low execution time of the decompression application. When communication devices with higher energy costs per transmitted byte are used, it would be preferable to use GZIP.

## 5 References

- [1] P. Deutsch. Deflate compressed data format specification version 1.3. RFC 1951, Aladdin Enterprises, May 1996.
- [2] Adam Dunkels, Niclas Finne, Joakim Eriksson, and Thiemo Voigt. Run-time dynamic linking for reprogramming wireless sensor networks. In *Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems (SenSys 2006)*, Boulder, Colorado, USA, November 2006. URL: <http://www.sics.se/~adam/dunkels06runtime.pdf>
- [3] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proc. SenSys'04*, Baltimore, Maryland, USA, November 2004.
- [4] J. Jeong and D. Culler. Incremental network programming for wireless sensors. In *Proceedings of the First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks IEEE SECON (2004)*, October 2004.
- [5] J. Koshy and R. Pandey. Remote incremental linking for energy-efficient reprogramming of sensor networks. In *Proceedings of the second European Workshop on Wireless Sensor Networks (EWSN) 2005*, 2005.
- [6] P. Levis and D. Culler. Maté: A tiny virtual machine for sensor networks. In *Proceedings of ASPLOS-X*, San Jose, CA, USA, October 2002.