A

On the Performance of Lossy Compression Schemes for Energy Constrained Sensor Networking

DAVIDE ZORDAN, University of Padova
BORJA MARTINEZ and IGNASI VILAJOSANA, Worldsensing
MICHELE ROSSI, University of Padova

Lossy temporal compression is key for energy constrained wireless sensor networks (WSN), where the imperfect reconstruction of the signal is often acceptable at the data collector, subject to some maximum error tolerance. In this paper, we evaluate a number of selected lossy compression methods from the literature, and extensively analyze their performance in terms of compression efficiency, computational complexity and energy consumption. Specifically, we first carry out a performance evaluation of existing and new compression schemes, considering linear, autoregressive, FFT-/DCT- and Wavelet-based models , by looking at their performance as a function of relevant signal statistics. Second, we obtain formulas through numerical fittings, to gauge their overall energy consumption and signal representation accuracy. Third, we evaluate the benefits that lossy compression methods bring about in interference-limited multi-hop networks, where the channel access is a source of inefficiency due to collisions and transmission scheduling. Our results reveal that the DCT-based schemes are the best option in terms of compression efficiency but are inefficient in terms of energy consumption. Instead, linear methods lead to substantial savings in terms of energy expenditure by, at the same time, leading to satisfactory compression ratios, reduced network delay and increased reliability performance.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Computational Complexity, Lossy Data Compression, Temporal Data Compression, Wireless Sensor Networks

ACM Reference Format:
Zordan, D., Martinez, B., Vilajosana, I. and Rossi, M. 2012. On the Performance of Lossy Compression Schemes for Energy Constrained Sensor Networking. *ACM Trans. Sensor Netw.* V, N, Article A (January YYYY), 34 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

In recent years, wireless sensors and mobile technologies have experienced a tremendous upsurge. Advances in hardware design and micro-fabrication have made it pos-

The work in this paper has been supported in part by the MOSAICS project, "MOnitoring Sensor and Actuator networks through Integrated Compressive Sensing and data gathering," funded by the University of Padova under grant no. CPDA094077 and by the European Commission under the 7th Framework Programme (SWAP project, GA 251557 and CLAM project, GA 258359). The work of Ignasi Vilajosana and Borja Martinez has been supported, in part, by Spanish grants PTQ-08-03-08109 and INN-TU-1558.
Author's addresses: Davide Zordan and Michele Rossi, Department of Information Engineering (DEI), University of Padova, Via Gradenigo 6/B, 35131 Padova, Italy; Borja Martinez and Ignasi Vilajosana, Worldsensing, Carrer Aragó 383, 08013 Barcelona, Spain.

DOI: http://dx.doi.org/10.1145/0000000.0000000

sible to potentially embed sensing and communication devices in every object, from banknotes to bicycles [Atzoria et al. 2010].

Wireless Sensor Network (WSN) technology has now reached a good level of maturity, as testified by the many emerging industrial standardization efforts [Palattella et al. 2012]. Notable WSN application examples include environmental monitoring [Szewczyk et al. 2004], geology [Werner-Allen et al. 2006] structural monitoring [Xu et al. 2004], smart grid and household energy metering [Kappler and Riegel 2004; Benzi et al. 2011]. These applications often require the collection and the subsequent analysis of large amounts of data, which are to be sent through suitable routing protocols to some data collection point(s). One of the main problems of this is related to the large number of devices: if this number will keep increasing as predicted in [Dodson 2003], and all signs point toward this direction, the amount of data to be managed by the network will become prohibitive. Further issues are due to the constrained nature of sensor nodes in terms of limited energy resources (devices are often battery operated) and to the fact that radio activities are their main source of energy consumption. This, together with the fact that sensor nodes are required to remain unattended and operational for long periods of time, poses severe constrains.

Several strategies have been developed to prolong the lifetime of sensor nodes. These comprise processing techniques such as data aggregation [Fasolo et al. 2007], distributed [Pattem and Krishnamachari 2004] or temporal [Sharaf et al. 2003] compression as well as battery replenishment through energy harvesting [Vullers et al. 2010]. The rationale behind data compression is that we can trade some additional energy for compression for some reduction in the energy spent for transmission. As we shall see in the remainder of this paper, this allows some important savings.

In this paper, we focus on the energy saving opportunities offered by data processing and, in particular, on the *lossy temporal compression* of data. With lossy techniques, the original data is compressed by however discarding some of the original information in it so that, at the receiver side, the decompressor can reconstruct the original data up to a certain accuracy. Lossy compression makes it possible to trade some reconstruction accuracy for some additional gains in terms of compression ratio with respect to lossless schemes. Note that these gains correspond to further savings in terms of transmission needs and that, depending on the application, some small inaccuracy in the reconstructed signal may be acceptable. Thus, lossy compression introduces some additional flexibility as one can tune the compression ratio as a function of energy consumption criteria.

We note that much of the existing literature has been devoted to the systematic study of lossless compression. [Marcelloni and Vecchio 2009] proposes a simple Lossless Entropy Compression (LEC) algorithm, comparing LEC with standard techniques such as gzip, bzip2, rar and classical Huffman and arithmetic encoders. A simple lossy compression scheme, called Lightweight Temporal Compression (LTC) [Schoellhammer et al. 2004], was also considered. However, the main focus of this comparison has been on the achievable compression ratio, whereas considerations on energy savings are only given for LEC. [van der Byl et al. 2009] examines Huffman, Run Length Encoding (RLE) and Delta Encoding (DE), comparing the energy spent for compression for these schemes. [Liang 2011] treats lossy (LTC) as well as lossless (LEC and Lempel-Ziv-Welch) compression methods, but only focusing on their compression performance. Further work is carried out in [Sadler and Martonosi 2006], where the energy savings from lossless compression algorithms are evaluated for different radio setups, in single- as well as multi-hop networks. Along the same lines, [Barr and Asanović 2006] compares several lossless compression schemes for a StrongArm CPU architecture, showing that data compression in some cases may cause

an increase in the overall energy expenditure. A comprehensive survey of practical lossless compression schemes for WSN can be found in [Srisooksai et al. 2012]. The lesson that we learn from these papers is that lossless compression can provide some energy savings. These are however smaller than one might expect because, for the sensor hardware in use nowadays, the energy spent for the execution of the compression algorithms (CPU) may be of the same order of magnitude of that spent for transmission (radio).

Further work has been carried out for what concerns lossy compression schemes. LTC [Lu et al. 2010], PLAMLiS [Liu et al. 2007] and the algorithm of [Pham et al. 2008] are all based on Piecewise Linear Approximation (PLA). Adaptive Auto-Regressive Moving Average (A-ARMA) [Lu et al. 2010] is based on ARMA models and RACE [Chen et al. 2004] exploits Wavelet-based compression. Also, [Marcelloni and Vecchio 2010] presents a lightweight compression framework based on Differential Pulse Coding Modulation (DPCM) where dictionaries are selected offline through multi-objective evolutionary optimization. Nevertheless, we remark that no systematic energy comparison has been carried out so far for lossy schemes. In this case, it is not clear whether lossy compression can be advantageous in terms of energy savings and what the involved tradeoffs are in terms of compression ratio *vs* representation accuracy and yet how these affect the overall energy expenditure. In addition, it is unclear whether linear and autoregressive schemes can provide any advantages at all compared to more sophisticated techniques such as Fourier- or Wavelet-based transforms, which have been effectively used to compress audio and video signals and for which fast and computationally efficient algorithms exist. In this paper, we fill these gaps by systematically comparing selected lossy temporal compression methods from the literature including polynomial, Fourier (FFT and DCT) and Wavelet compression schemes. We remark that alternative approaches, such as data aggregation [Fasolo et al. 2007] are also possible. However, these are out of the scope of our current investigation in this paper, which focuses on the temporal and lossy compression of time series.

The main contributions of this paper are:

— We consider selected lossy compression algorithms for time series, accounting for linear (e.g., LTC [Lu et al. 2010]), autoregressive (e.g., A-ARMA [Lu et al. 2010]) models, Fourier and Wavelet transforms. At first, we focus on interference-free single- and multi-hop networks, where the Medium Access Control (MAC) layer is idealized, i.e., besides transmission and reception, it does not introduce further energetic inefficiencies due to collisions and idle times for floor acquisition. For this scenario, we assess whether signal compression actually helps in the reduction of the overall energy consumption, depending on the compression algorithm, the chosen reconstruction fidelity, the signal statistics and the hardware characteristics.
— We provide formulas, obtained through numerical fittings and validated against real datasets, to gauge the computational complexity, the overall energy consumption and the signal representation accuracy of the best performing compression algorithms as a function of the most relevant system parameters. These formulas can be used to generalize the results obtained here to other WSN architectures.
— We consider interference-limited multi-hop networks where multiple nodes contend for the channel and data traverses a data collection tree until it reaches a data collection point located at its root (the WSN “sink”). Thus, we analytically characterize this second scenario by evaluating the performance improvements that are brought about by different lossy compression schemes in the presence of collisions and idle times for floor acquisition at the MAC.

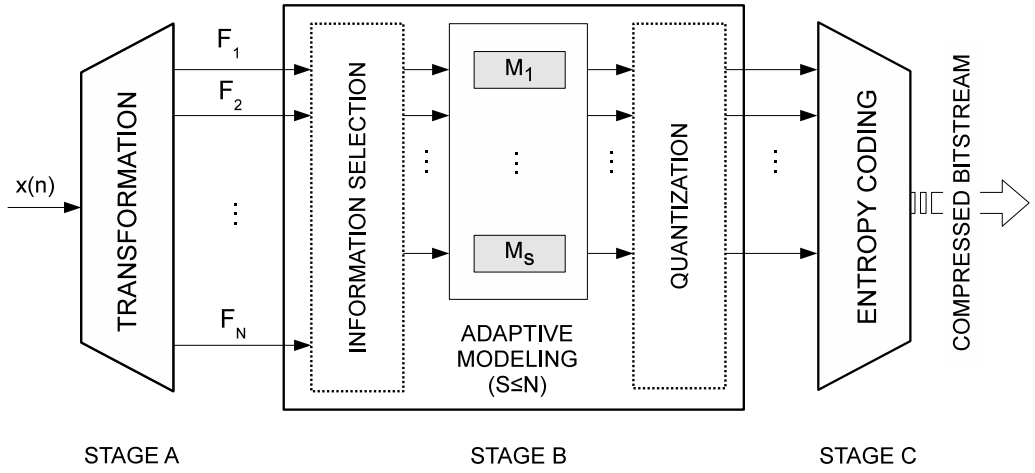


Fig. 1. General lossy compression diagram.

The rest of the paper is organized as follows. In Section 2 we discuss selected lossy compression algorithms from the literature, along with some lossy compression schemes that we introduce in this paper. In Sections 3 and 4 we carry out our performance evaluation of lossy compression for the interference-free and the interference-limited WSN scenarios, respectively. We finally draw our conclusions in Section 5.

2. LOSSY COMPRESSION FOR CONSTRAINED SENSOR NODES

To facilitate the description of the compression schemes considered in this paper and to identify their essential features, in Fig. 1 we show the diagram of a generic lossy compression algorithm, see, e.g., [Wallace 1992]. The following three fundamental stages are identified:

A **Transformation:** this stage entails the representation of the input signal (time series $x(n)$) into a convenient transformation domain. That is, the signal is decomposed into a number N of coefficients $\{F_1, \ldots, F_N\}$ in the new domain. As an example, FFT, DCT and Wavelet transforms represent time series into the frequency domain.

B **Adaptive modeling:** a number of coefficients $S \leq N$ is selected so that these will be sufficient to represent the signal within a certain target accuracy. Moreover, a further adaptive modeling phase (models $\{M_1, \ldots, M_S\}$) can be applied on the time series corresponding to each of the selected coefficients and, finally, a quantizer can be employed to represent the data through a finite number of levels.

C **Entropy coding:** the quantized data can be encoded using an *entropy coder* (EC) to obtain additional compression. Entropy represents the amount of information present in the data, and an EC encodes the given set of symbols with the minimum number of bits required to represent them.

As a popular example, JPG image compression [Wallace 1992] matches this model as follows: Stage-A: DCT, Stage-B: DPCM modeling for the DC coefficients, quantization for all coefficients, with run length encoding for null coefficients after quantization, Stage-C: huffman coding (arithmetic coding is also supported).

We remark that a specific compression algorithm does not necessary have to implement all the three stages above, but some of them can be omitted or only partially taken into account. For example, for Stage-B we could use the selection and quantization blocks, without any adaptive modeling. In WSNs, the exact combination of algo-

rithms to use depends on the reconstruction accuracy goal as well as on the affordable computational complexity.

In the following, we briefly review the lossy signal compression methods that will be characterized in this paper. Due to the contained nature of the sensor devices, these schemes only use some of the above stages. In Section 2.1, we discuss techniques based on Fourier and Wavelet transforms (Stage-A). In Section 2.2, we describe adaptive modeling techniques (Stage-B). Finally, in Section 2.3 we discuss a lightweight scheme based on quantization and entropy coding (Stage-C).

2.1. Compression Methods Based on Fourier and Wavelet Transforms (Stage-A)

For these techniques, compression is achieved through sending subsets of the FFT, DCT or Wavelet transformation coefficients. We came up with some possible methods, presented below, that differ in how the transformation coefficients are picked. These algorithms first transform the signal into a suitable domain (Stage-A) and subsequently use the *information selection* block of Stage-B.

2.1.1. Fast Fourier Transform (FFT). The first method that we consider relies on the simplest way to use the Fourier transform for compression. Specifically, the input time series $x(n)$ is mapped to its frequency representation $X(f) \in \mathbb{C}$ through a Fast Fourier Transform (FFT). We define $X_{\mathcal{R}}(f) \triangleq \mathfrak{Re}\{X(f)\}$, and $X_{\mathcal{I}}(f) \triangleq \mathfrak{Im}\{X(f)\}$ as the real and the imaginary part of $X(f)$, respectively. Since $x(n)$ is a real-valued time series, $X(f)$ is Hermitian, i.e., $X(-f) = \overline{X(f)}$. This symmetry allows the FFT to be stored using the same number of samples N of the original signal. For N even we take $f \in \{f_1, \ldots, f_{N/2}\}$ for both $X_{\mathcal{R}}(\cdot)$ and $X_{\mathcal{I}}(\cdot)$, while if N is odd we take $f \in \{f_1, \ldots, f_{\lfloor N/2 \rfloor + 1}\}$ for the real part and $f \in \{f_1, \ldots, f_{\lfloor N/2 \rfloor}\}$ for the imaginary part.

The compressed representation $\hat{X}(f) \triangleq \hat{X}_{\mathcal{R}}(f) + j\hat{X}_{\mathcal{I}}(f)$ will also be in the frequency domain and it is built (for the case of N even) as follows:

(1) initialize $\hat{X}_{\mathcal{R}}(f) = 0$ and $\hat{X}_{\mathcal{I}}(f) = 0, \forall\, f \in \{f_1, \ldots, f_{N/2}\}$;
(2) select the coefficient with maximum absolute value from $X_{\mathcal{R}}$ and $X_{\mathcal{I}}$, i.e., $f_{\max} \triangleq \operatorname{argmax}_f \max\{|X_{\mathcal{R}}(f)|, |X_{\mathcal{I}}(f)|\}$ and $M \triangleq \operatorname{argmax}_{i \in \{\mathcal{R}, \mathcal{I}\}}\{|X_i(f_{\max})|\}$;
(3) set $\hat{X}_M(f_{\max}) = X_M(f_{\max})$ and then set $X_M(f_{\max}) = 0$;
(4) if $\hat{x}(n)$, the inverse FFT of $\hat{X}(f)$, meets the error tolerance constraint continue, otherwise repeat from step (2);
(5) encode the values and the positions of the harmonics stored in $\hat{X}_{\mathcal{R}}$ and $\hat{X}_{\mathcal{I}}$.

Hence, the decompressor at the receiver obtains $\hat{X}_{\mathcal{R}}(f)$ and $\hat{X}_{\mathcal{I}}(f)$ and exploits the Hermitian symmetry to reconstruct $\hat{X}(f)$.

Note that the above coefficient selection method resembles a K non-linear approximation, as usually implemented by image processing techniques see, e.g., [Donoho et al. 1998]. In our case, K (the number of coefficients to be retained) is dynamically selected depending on the input signal characteristics. We emphasize that alternative schemes for the selection of the Fourier coefficients are also possible. For instance, one may select the FFT coefficients based on the maximum absolute magnitude of their complex values and then retain both the real and imaginary part of the selected coefficients. We found marginal differences among the various approaches.

2.1.2. Low Pass Filter (FFT-LPF). We implemented a second FFT-based lossy algorithm, which we have termed FFT-LPF. Since the input time series $x(n)$ is a slowly varying signal in many common cases (i.e., having strong temporal correlation) with some high frequency noise superimposed, most of the significant coefficients of $X(f)$ reside in the

low frequencies. For FFT-PLF, we start setting $\hat{X}(f) = 0$ for all frequencies. Thus, $X(f)$ is evaluated from f_1, incrementally moving toward higher frequencies, $f_2, f_3, \ldots$. At each iteration i, $X(f_i)$ is copied onto $\hat{X}(f_i)$ (both real and imaginary part), the inverse FFT is computed taking $\hat{X}(f)$ as input and the error tolerance constraint is checked on the so obtained $\hat{x}(n)$. If the given tolerance is met the algorithm stops, otherwise it is reiterated for the next frequency f_{i+1}.

Note that this method resembles a K linear approximation scheme, where the selection order is fixed (LPF), but the number of coefficients to be retained, K, is dynamically adjusted in order to meet a given error tolerance.

2.1.3. Windowing. The two algorithms discussed above suffer from an edge discontinuity problem. In particular, when we take the FFT over a window of N samples, if $x(1)$ and $x(N)$ differ substantially the information about this discontinuity is spread across the whole spectrum in the frequency domain. Hence, in order to meet the tolerance constraint for all the samples in the window, a high number of harmonics is selected by the previous algorithms, resulting in a poor compression and in a high number of operations.

To solve this issue, we implemented a version of the FFT algorithm that considers overlapping windows of $N+2W$ samples instead of disjoint windows of length N, where W is the number of samples that overlap between subsequent windows. The first FFT is taken over the entire window and the selection of the coefficients goes on depending on the selected algorithm (either FFT or FFT-LPF), but the tolerance constraint is only checked on the N samples in the central part of the window. With this workaround we can get rid of the edge discontinuity problem and encode the information about the N samples of interest with very few coefficients as it will be seen shortly in Section 3. As a drawback, the direct and inverse transforms have to be taken on longer windows, which results in a higher number of operations.

2.1.4. Discrete Cosine Transform (DCT). We also considered the Discrete Cosine Transform (type II), mainly for three reasons: 1) its coefficients are real, so we did not have to cope with real and imaginary parts, thus saving memory and number of operations; 2) it has a strong "energy compaction" property [Rao and Yip 1990], i.e., most of the signal information tends to be concentrated in a few low-frequency components; 3) the DCT of a signal with N samples is equivalent to a DFT on a real signal of even symmetry with double length, so the DCT does not suffer from the edge discontinuity problem.

2.1.5. Wavelet Transform (WT). As an alternative to Fourier schemes, several methods based upon multi-resolution analysis have been proposed in the literature. RACE [Chen et al. 2004] is a notable example: it features a compression algorithm based on the Fast Wavelet Transform (FWT) of the signal (Stage-A) followed by the selection of a number of coefficients (Stage-B) that are used to represent the input signal within given error bounds. As for DCT schemes, the compression mainly takes place in the selection step.

In [Chen et al. 2004], a Haar basis function is used for the wavelet decomposition step. The most remarkable contribution of RACE is the way in which the wavelet coefficients are selected. Most traditional compression algorithms, after the FWT, just pick the largest coefficients, i.e., the selection step is based on a threshold value, whereby all the coefficients below this threshold are discarded, whereas those above it are retained. Differently, in RACE, the Haar wavelet coefficients are arranged into a tree structure. Then, thanks to some special properties of the Haar functions, at each node of the tree, the error in the reconstruction of the signal is estimated assuming that this node (i.e., the corresponding coefficient) and all its children in the tree are omitted.

This selection method has two important properties. First, the signal representation error can be evaluated on-the-fly during the decomposition and the maximum error tolerance can be maintained under control, without having to compute any inverse wavelet transform.[1] Second, compression can be achieved in an incremental way, by descending the tree and adding nodes until the desired precision is reached (of course, the higher the number of coefficients, the lower the compression performance). These facts are very important for energy constrained WSNs and, as we will see in Section 3, lead to a smaller energy for compression with respect to DCT and FFT schemes.

2.2. Compression Methods Based on Adaptive Modeling (Stage-B)

In Adaptive Modeling schemes, some signal model is iteratively updated over time, exploiting the correlation structure of the signal through linear, polynomial or autoregressive methods. Specifically, the input time series is collected and processed according to transmission windows of N samples each. At the end of each time window the selected compression method is applied, obtaining a set of model parameters that are transmitted in place of the original data. In the adaptive modeling schemes described below, information selection is not used, as they do not employ any transformation stage.

2.2.1. Piecewise Linear Approximations (PLA). The idea of PLA is to use a sequence of line segments to represent an input time series $x(n)$ over pre-determined time windows (of N samples) with a bounded approximation error. For most time series consisting of environmental measures, linear approximations work well enough over short time frames. Further, since a line segment can be determined by only two end points, PLA leads to quite efficient implementations in terms of memory and transmission requirements.

The approximated signal is hereafter referred to as $\hat{x}(n)$, the error with respect to the actual value is given by the Euclidean distance $|\hat{x}(n) - x(n)|$. Most PLA algorithms use standard least squares fitting to calculate the approximating line segments. Often, a further simplification is introduced to reduce the computational complexity, which consists of forcing the end points of each line segment to be points of the original time series $x(n)$. This makes least squares fitting unnecessary as the line segments are fully identified by the extreme points of $x(n)$ in the considered time window. The following schemes exploit this approach.

Lightweight Temporal Compression (LTC) [Schoellhammer et al. 2004]: the LTC algorithm is a low complexity PLA technique. Specifically, let $x(n)$ be the points of a time series with $n = 1, 2, \ldots, N$. The LTC algorithm starts with $n = 1$ and fixes the first point of the approximating line segment to $x(1)$. The second point $x(2)$ is transformed into a vertical line segment that determines the set of all "acceptable" lines $\Omega_{1,2}$ with starting point $x(1)$. This vertical segment is centered at $x(2)$ and covers all values meeting a maximum tolerance $\varepsilon \geq 0$, i.e., lying within the interval $[x(2) - \varepsilon, x(2) + \varepsilon]$, see Fig. 2(a). The set of acceptable lines for $n = 3$, $\Omega_{1,2,3}$, is obtained by the intersection of $\Omega_{1,2}$ and the set of lines with starting point $x(1)$ that are acceptable for $x(3)$, see Fig. 2(b). If $x(3)$ falls within $\Omega_{1,2,3}$ the algorithm continues with the next point $x(4)$ and the new set of acceptable lines $\Omega_{1,2,3,4}$ is obtained as the intersection of $\Omega_{1,2,3}$ and the set of lines with starting point $x(1)$ that are acceptable for $x(4)$. The procedure is iterated adding one point at a time until, at a given step s, $x(s)$ is not contained in $\Omega_{1,2,\ldots,s}$. Thus, the algorithm sets $x(1)$ and $x(s-1)$ as the starting and ending points of

[1]Note that in the FFT and DCT methods of above, the error tolerance check always entails the computation of an inverse transformation at the source.

the approximating line segment for $n = 1, 2, \ldots, s-1$ and starts over with $x(s-1)$ considering it as the first point of the next approximating line segment. In our example, $s = 4$, see Fig. 2(b).

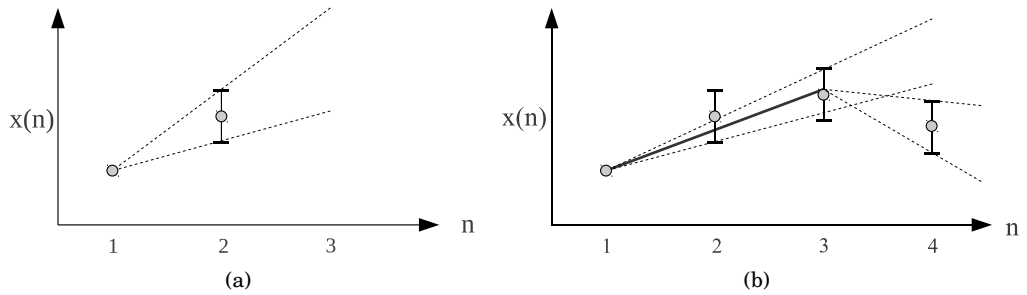


Fig. 2. Lightweight Temporal Compression example.

When the inclusion of a new sample does not comply with the allowed maximum tolerance, the algorithm starts over looking for a new line segment. Thus, it self-adapts to the characteristics of $x(n)$ without having to fix beforehand the lapse of time between subsequent updates.

PLAMLiS [Liu et al. 2007]: as LTC, PLAMLiS represents the input data series $x(n)$ through a sequence of line segments. Here, the linear fitting problem is converted into a set-covering problem, trying to find the minimum number of segments that cover the entire set of values over a given time window. This problem is then solved through a greedy algorithm as explained in [Liu et al. 2007]. This algorithm is outperformed in terms of complexity by its enhanced version, that we discuss next.

Enhanced PLAMLiS [Pham et al. 2008]: is a top-down recursive segmentation algorithm with smaller computational cost with respect to PLAMLiS. Consider the input time series $x(n)$ and a time window $n = 1, 2, \ldots, N$. The algorithm starts by taking a first segment $(x(1), x(N))$, if the maximum allowed tolerance ε is met for all points along this segment the algorithm ends. Otherwise, the segment is split in two segments at the point $x(i)$, $1 < i < N$, where the error is maximum, obtaining the two segments $(x(1), x(i))$ and $(x(i), x(N))$. The same procedure is recursively applied on the resulting segments until the maximum error tolerance is met for all points.

2.2.2. Polynomial Regression (PR). The above methods can be modified by relaxing the constraint that the endpoints of the segments $x(i)$ and $x(j)$ ($j > i$) must be actual points of $x(n)$. In this case, polynomials of given order $p \geq 1$ are used as the approximating functions, whose coefficients are found through standard regression methods based on least squares fitting [Phillips 2003]. Specifically, we start with a window of p samples, since a p-order polynomial exactly interpolates p points, for which we obtain the best fitting polynomial coefficients. Thus, we keep increasing the window length of one sample at a time, computing the new coefficients, and stop when the target error tolerance is no longer met.

We remark that, tracing a line between two fixed points as done by LTC and PLAMLiS has a very low computational complexity, while least squares fitting can have a significant cost. Polynomial regression obtains better results in terms of approximation at the cost of higher computational complexity (which increases with the polynomial order).

2.2.3. Auto-Regressive (AR) Methods. Auto Regressive (AR) models in their multiple flavors (AR, ARMA, ARIMA, etc.) have been widely used for time series modeling and forecasting in fields like macro-economics or market analysis. The basic idea is to obtain a model based on the history of the sampled data, i.e., on its correlation structure. When used for signal compression, AR obtains a model from the input data and sends this model to the receiver in place of the actual time series. The reconstructed model is thus used at the data collection point (the sink) for data prediction until it is updated by the encoder device. Specifically, each node locally verifies the accuracy of the predicted data values with respect to the collected samples. If the accuracy is within a prescribed error tolerance, the node assumes that the current model will be sufficient for the sink to rebuild the data within the given error tolerance. Otherwise, the parameters from the current model are encoded and a new model is built as a replacement for the old one. As said above, the model parameters are sent to the sink at the end of each transmission window in place of the original data.

Adaptive Auto-Regressive Moving Average (A-ARMA) [Lu et al. 2010]: the basic idea of A-ARMA [Lu et al. 2010] is that of having each sensor node compute an ARMA model based on $N' < N$ consecutive samples. In order to reduce the complexity in the model estimation process, adaptive ARMA employs low-order models, whereby the validity of the model being used is checked through a moving window technique. Specifically, a sensor node builds an ARMA model $M^{(0)} = ARMA(p, q, N', 0)$ considering N' samples starting from the first sample (sample 0) of the current transmission window (p and q are the orders related to the auto-regressive and moving average components of the ARMA filter). Hence, this model is updated considering N' subsequent samples at a time until the prescribed error tolerance is met, at which point a new ARMA model is built and the update/check procedure is iterated for this one. At the end of the transmission window of N samples, the parameters of all the ARMA models that have been obtained to describe the input time series (within the prescribed error tolerance) are sent to the sink in place of the original data, as discussed above.

Modified Adaptive Auto-Regressive (MA-AR): according to A-ARMA the model is updated over fixed-size windows of N' samples. A drawback of this is that, especially for highly noisy environments, the estimation over fixed-size windows can lead to poor results when used for forecasting. MA-AR allows the estimation to be performed on time windows whose size is adapted according to the signal statistics. A more detailed discussion of ARMA methods can be found in [Zordan et al. 2012].

2.3. Compression Methods Based on Entropy Coding (Stage-C)

As a representative technique for Stage-C we consider the algorithm in [Marcelloni and Vecchio 2010], proposed by Marcelloni and Vecchio (MV). This algorithm works in three steps: (a) Differential Pulse-Modulation Coding (DPCM), (b) quantization and (c) huffman entropy encoding. After de-noising, step (a) employs a simple differential encoding model (DPCM), which operates on the differences between consecutive input samples. The rationale behind this differential scheme is that WSN signals are usually smooth and slow time-varying. Hence, the difference between samples is expected to be small, leading to a small amount of information to be encoded.

In the quantization block (b), the difference between subsequent samples is quantized. This is the most important step of the algorithm and probably where most of the compression performance is achieved. In fact, given the small expected value of the DPCM differences, a quantizer with only a small number of levels can be used without impacting too much on the signal representation accuracy.

In our performance evaluation, in order to carry out a fair comparison among the considered compression schemes, we bound the maximum error tolerance for each sample, setting it as a constant input parameter equal for all the algorithms. As we did for the other compression schemes, the MV algorithm has been as well adapted to consider this. Specifically, a first pass is performed to find the maximum difference at the output of the DPCM. Based on this, the number of levels of the quantizer is selected so that the quantization error remains smaller than a target error tolerance; this returns the quantizer for the given input signal. After this, a second pass is executed, using the selected quantizer, to obtain the final encoded symbols. Note that this is slightly different from [Marcelloni and Vecchio 2010], where optimal quantizers are calculated offline through a dedicated optimization stage following different optimization criteria. While the latter approach is also valuable, it does not allow for a precise control of the maximum error tolerance and a fair comparison with the other compression schemes that we consider in this paper.

Finally, the entropy encoding step (c) exploits the fact that the quantization levels have different probabilities. Once again, environmental signals are quite smooth and therefore small differences are more likely. Hence, a Huffman encoder is designed to assign the shorter binary codewords to the most probable levels. The set of binary codewords is selected so that no member is a prefix of another member and, in turn, the corresponding code is uniquely decodable. This dictionary can be sent together with the compressed data frame, or can be statistically precomputed and shared between the communicating entities.

3. PERFORMANCE COMPARISON FOR INTERFERENCE-FREE NETWORKS

This section focuses on single- and multi-hop WSNs where the interference due to channel access is negligible or absent. In this case, the energy expenditure at the MAC is only confined to transmission and reception energy, by also keeping into account the protocol overhead at the MAC in terms of packet headers. However, further energetic inefficiencies due to channel contentions and waiting times due to floor acquisition are neglected (their impact will be considered later on in Section 4). The objectives of this section are:

— to provide a thorough performance comparison of the compression methods of Section 2. The selected performance metrics are: 1) compression ratio, 2) computational and transmission energy and 3) reconstruction error at the receiver, which will be defined below.
— To quantify the impact on the compression performance of the statistical properties of the input signals.
— To investigate whether or not data compression leads to energy savings in single- and multi-hop interference-free WSN scenarios, and obtain quantitative measurements of possible benefits as a function of compression ratio and energy consumption of the wireless end-nodes hardware (micro-controller and radio).
— To obtain, through numerical fitting, close-form equations which model the considered performance metrics as a function of key parameters.

Toward the above objectives, we present simulation results obtained using synthetic signals with varying correlation length. These signals make it possible to give a fine grained description of the performance of the selected techniques, so as to look comprehensively at the entire range of variation of the temporal correlation statistics. Real datasets are then used to validate the proposed empirical fitting formulas.

3.1. Preliminary Definitions

Before delving into the description of the results, in the following we give some definitions.

Definition 3.1. Correlation length
Given a stationary discrete time series $x(n)$ with $n = 1, 2, \ldots, N$, we define **correlation length** of $x(n)$ as the smallest value $n^\star$ such that the autocorrelation function of $x(n)$ is smaller than a predetermined threshold $\rho_{\rm th}$. The autocorrelation is:

$$\rho_x(n) = \frac{\mathrm{E}\left[(x(m) - \mu_x)(x(m+n) - \mu_x)\right]}{\sigma_x^2} \ ,$$

where μ_x and σ_x^2 are the mean and the variance of $x(n)$, respectively. Formally, $n^\star$ is defined as:

$$n^\star = \underset{n>0}{\operatorname{argmin}} \left\{\rho_x(n) < \rho_{\rm th}\right\} \ .$$

Below, we define the performance metrics that will be considered in the remainder of this paper.

Definition 3.2. Compression ratio
Given a finite time series $x(n)$ and its compressed version $\hat{x}(n)$, we define **compression ratio** η the quantity:

$$\eta = \frac{N_b(\hat{x})}{N_b(x)} \ ,$$

where $N_b(\hat{x})$ and $N_b(x)$ are the number of bits used to represent the compressed time series $\hat{x}(n)$ and the original one $x(n)$, respectively.

Definition 3.3. Reconstruction error and error tolerance
Given a discrete time series $x(n)$ and its compressed version $\hat{x}(n)$, we define the reconstruction error at time $n \geq 1$ as $e(n) = |x(n) - \hat{x}(n)|$, where $|\cdot|$ is the Euclidean distance. The error tolerance ε is the maximum permitted error at the receiver, i.e., it must be $e(n) \leq \varepsilon$ for all n.

Definition 3.4. Energy consumption for compression
Is the energy drained from the battery to accomplish the compression task. For every compression method we have recorded the number of operations to process the original time series $x(n)$ accounting for the number of additions, multiplications, divisions and comparisons. Thus, depending on selected hardware architecture, we have mapped these figures into the corresponding number of clock cycles and we have subsequently mapped the latter into the corresponding energy expenditure.

Definition 3.5. Transmission Energy
Is the energy consumed for transmission, obtained accounting for the radio chip characteristics, channel attenuation effects and the protocol overhead due to physical (PHY) and medium access (MAC) layers.

Definition 3.6. Total Energy Consumption
Is the sum of the energy consumption for compression and transmission and is expressed in [Joule].

In the computation of the energy consumption for compression, we only accounted for the operations performed by the CPU, without considering the possible additional costs related to other peripherals of the micro-controller.

For the communication cost we have only taken into consideration the transmission energy, neglecting the cost of switching the radio transceiver on and off and the energy spent at the destination to receive the data. The former are fixed costs that would also be incurred without compression, while the latter can be ignored if the receiver is not a power constrained device. Moreover, we do not consider link-level retransmissions due to channel errors or multi-user interference.

3.2. Generation of Synthetic Stationary Signals

The synthetic stationary signals have been obtained through a known method to enforce the first and second moments to a white random process, see [Davies and Harte 1987][Zordan et al. 2011]. Our objective is to obtain a random time series $x(n)$ with given mean μ_x, variance σ_x^2 and autocorrelation function $\rho_x(n)$. The procedure works as follow:

(1) A random Gaussian series $G(k)$ with $k = 1, 2, \ldots, N$ is generated in the frequency domain, where N is the length of the time series $x(n)$ that we want to obtain. Every element of $G(k)$ is an independent Gaussian random variable with mean $\mu_G = 0$ and variance $\sigma_G^2 = 1$.
(2) The Discrete Fourier Transform (DFT) of the autocorrelation function $\rho_x(n)$ is computed, $S_x(k) = \mathcal{F}[\rho_x(n)]$, where $\mathcal{F}[\cdot]$ is the DFT operator.
(3) We compute the entry-wise product $X(k) = G(k) \circ S_x(k)^{\frac{1}{2}}$.
(4) We finally obtain the correlated and Gaussian time series $x(n)$ as $\mathcal{F}^{-1}[X(k)]$.

This is equivalent to filter a white random process with a linear, time invariant filter, whose transfer function is $\mathcal{F}^{-1}[S_x(k)^{\frac{1}{2}}]$. The stability of this procedure is ensured by a suitable choice for the correlation function, which must be square integrable. For the simulations in this paper we have used a Gaussian correlation function [Abrahamsen 1997], i.e., $\rho_x(n) = \exp\{-an^2\}$, where a is chosen in order to get the desired correlation length $n^\star$ as follows:

$$a = -\frac{\log(\rho_{\text{th}})}{(n^\star)^2} \ .$$

Without loss of generality, we generate synthetic signals with $\mu_x = 0$ and $\sigma_x^2 = 1$. In fact, applying an offset to the generated signals and a scale factor does not change the resulting correlation. For an in depth characterization of the Gaussian correlation function see [Abrahamsen 1997].

Also, to emulate the behavior of real WSN signals, we superimpose noise to the synthetic signals, so as to mimic random perturbations due to limited precision of the sensing hardware and random fluctuations of the observed physical phenomenon. The noise is modeled as a zero mean white Gaussian process with standard deviation σ_{noise}.

3.3. Hardware Architecture

We selected the TI MSP430 [Bierl 2000] micro-controller using the corresponding 16 bit floating point package for the calculations and for the data representation. In the active state, the MSP430 is powered by a current of 330 μA at 2.2 V and it has a clock rate of 1 MHz. The resulting energy consumption per CPU cycle is $E_0 = 0.726$ nJ. The number of clock cycles needed for the floating point operations are given in Table 5.8 of [Bierl 2000].

For radio, we selected the TI CC2420 RF transceiver [Chipcon 2007], an IEEE 802.15.4 [IEEE P802.15 Working Group 2003] compliant radio. For commercial radio transceivers, the current consumption associated with the transmission activity is typically selected from a finite set of values, that for the CC2420 are 8, varying from

a minimum of 8.5 mA to a maximum of 17.4 mA, with a supply voltage of 3.3 V for an effective data rate of 250 kbps, see [Chipcon 2007]. Thus, the energy cost associated with the transmission of a bit, $E'_{Tx}[\ell]$, given the current power level $\ell \in \{1, \ldots, 8\}$ ranges from 112 nJ to 230 nJ, which correspond to the energy spent by the micro-processor during 154 and 316 clock cycles, respectively. The current level, and consequently the output power of the radio transceiver, has to be chosen according to the considered scenario, which includes the transmission distance, the channel noise level, the type of environment (e.g., free space, indoor, presence of obstacles), etc.

We remark that the results that we obtain for this specific architecture can be promptly generalized to different CPUs and radios. As we show later in the paper, this is possible by separating algorithm-dependent and hardware-dependent terms in the calculation of the overall energy consumption. In particular, the compression performance of all algorithm is evaluated using 16 bits arithmetics, the natural choice for the MSP430, a 16-bit word processor.

3.4. Theoretical Bound for Signal Compression

Given the discrete and Gaussian time series of Section 3.2, from the theory in [Berger 1971] we can derive the theoretical lower bound on the transmission rate $R_{\min}$ (bits/sample):

$$R_{\min}(n^\star, N, \varepsilon) = \frac{1}{N} \sum_{i=1}^{N} \max \left\{ 0, \frac{1}{2} \log_2 \left(\frac{\zeta_i^2}{\varepsilon} \right) \right\} ,$$

where ε is the maximum permitted distortion at the receiver, N is the number of input samples and ζ_i are the eigenvalues of the covariance matrix $\mathbf{\Sigma}(n^\star)$ of $x(n)$ for $n = 1, \ldots, N$. Hence, for given $(n^\star, N, \varepsilon)$ we can bound the compression ratio achievable by any practical scheme as:

$$\eta \geq \frac{R_{\min}(n^\star, N, \varepsilon)}{R_0} , \tag{1}$$

where $R_0 = 16$ is the rate expressed in bits/sample in the uncompressed case for our hardware.

3.5. Simulation Setup

For the results that we discuss in what follows, we used synthetic signals with correlation length $n^\star$ varying in $\{1, 10, 20, 50, \ldots, 500\}$ samples, where after 20, $n^\star$ varies in steps of 30 (we have picked $\rho_{\text{th}} = 0.05$ for all the results shown in this paper). We consider time series of $N = 500$ samples (time slots) at a time, progressively taken from a longer realization of the signal, so as to avoid artifacts related to the generation technique. We recall that the signals are correlated Gaussian with zero-mean and unit variance. Moreover, a Gaussian noise with standard deviation $\sigma_{\text{noise}} = 0.04$ has been added to the signal, as per the signal generation method of Section 3.2. For the reconstruction accuracy, the absolute error tolerance has been set to $\varepsilon = \xi \sigma_{\text{noise}}$, with $\xi \geq 0$. In the following graphs, each point is obtained by averaging the outcomes of 10^4 simulation runs. For a fair comparison, the same realization of the input time series $x(n)$ has been used for all the compression methods considered, for each simulation run and value of $n^\star$. Moreover, all the compression algorithms have been configured with the same error tolerance, so that the energy compression and consumption figures that we obtain are for the same reconstruction fidelity at the receiver.

3.6. Compression Ratio vs Processing Energy

In the following, we analyze the performance in terms of compression effectiveness and computational complexity (energy) for the lossy compression methods of Section 2.

Adaptive Modeling Methods: in this first set of results we compare the performance of the following compression methods: 1) Modified Adaptive Autoregressive (M-AAR); 2) Polynomial Regression (PR); 3) Piecewise Linear Approximation (PLAMLiS); 4) Enhanced Piecewise Linear Approximation (E-PLAMLiS); 5) Lightweight Temporal Compression (LTC) and 6) Marcelloni and Vecchio's algorithm (MV) . For the M-AAR autoregressive filter and the polynomial regression (PR) we show results for the two orders, $p = \{2, 4\}$. The lower bound on the compression ratio η is also plotted for comparison, see (1).

Fig. 3(a) shows the Compression Ratio achieved by the six compression methods as a function of the correlation length $n^\star$. These results reveal that for small values of $n^\star$ the compression performance is poor for all compression schemes, whereas it improves for increasing correlation length, by reaching a floor value for sufficiently large $n^\star$. This confirms that $n^\star$ is a key parameter for the performance of all schemes. Also, the compression performance differs among the different methods, with PR giving the best results. This reflects the fact that, differently from all the other methods, PR approximates $x(n)$ without requiring its fitting curves to pass through the points of the given input signal. This entails some inherent filtering, that is embedded in this scheme and makes it more robust against small and random perturbations.

Fig. 3(b) shows the energy consumption for compression. For increasing values of $n^\star$ the compression ratio becomes smaller for all schemes, but their energy expenditure substantially differs. Notably, the excellent compression capabilities of PR are counterbalanced by its demanding requirements in terms of energy. M-AAR and PLAMLiS also require a quite large amount of processing energy, although this is almost one order of magnitude smaller than that of PR. LTC, E-PLAMLiS and MV have the smallest energy consumption among all schemes.

We now discuss the dependence of the computational complexity (which is strictly related to the energy spent for compression) on $n^\star$. LTC encodes the input signal $x(n)$ incrementally, starting from the first sample and adding one sample at a time. Thus, the number of operations that it performs only weakly depends on the correlation length and, in turn, the energy that it spends for compression is almost constant with varying $n^\star$. E-PLAMLiS takes advantage of the increasing correlation length: as the temporal correlation increases, this method has to perform fewer "divide and reiterate" steps, so the number of operations required gets smaller and, consequently, also the energy spent for compression is reduced. MV performs almost the same number of operations for different correlation lengths, except for very small values of $n^\star$. This occurs because, in order to meet the error constraint for uncorrelated signals ($n^\star \approx 1$), the quantization step has to use a high number of levels (DPCM signals have wider ranges), and with an increasing number of levels the entropy encoder assigns an exponentially increasing number of bits to some symbols. As a consequence, also the number of operations related to the assignment of these codewords increases. We recall that, in order to fairly compare MV with the other methods that we analyze in this paper, we adapted it as explained in Section 2.3. Specifically, in the evaluation of the energy consumption associated with the compression operation at the transmitter, we also consider the operations performed for the online selection of the quantizer, so as to meet a target error tolerance. In our case, the results differ from those in [Marcelloni and Vecchio 2010] where optimal quantizers are computed offline, and only the final encoding stage is performed on the nodes, which entails a lower

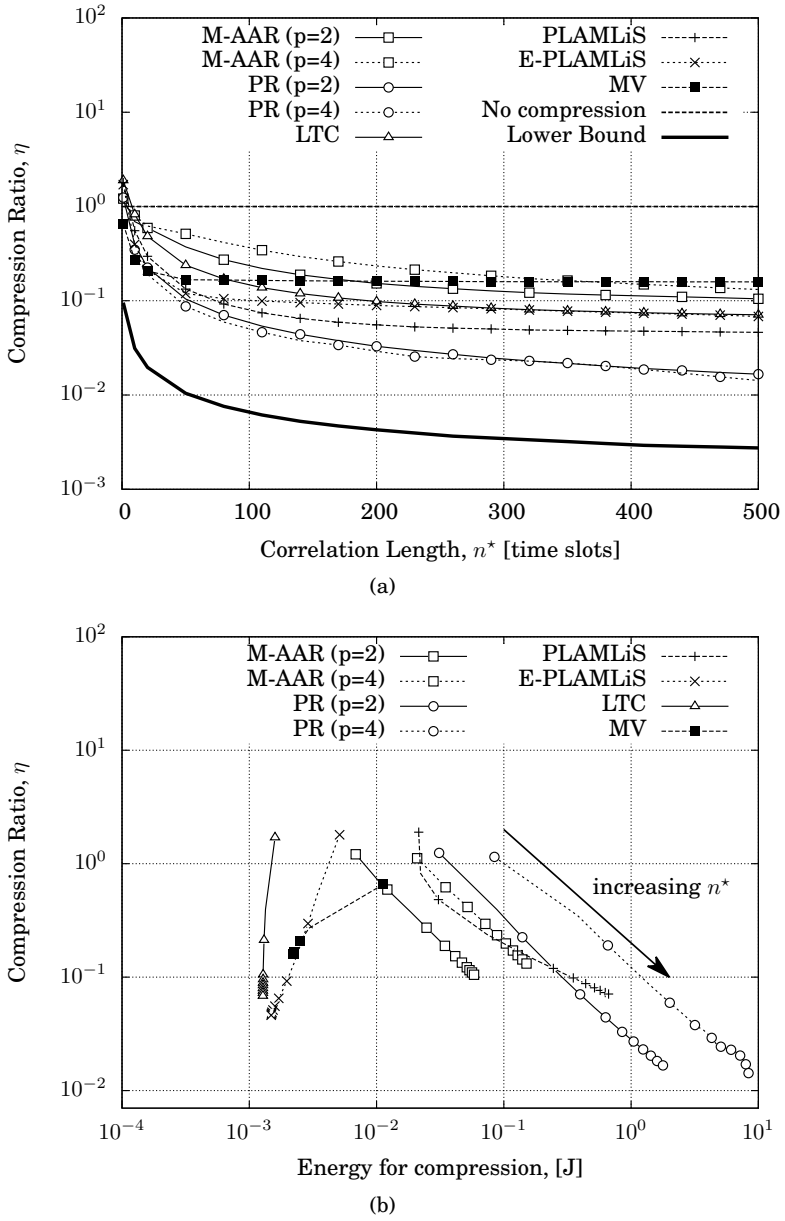


Fig. 3. (a) η *vs* Correlation Length $n^\star$ and (b) η *vs* Energy consumption for compression for the Adaptive Modeling methods for fixed $\varepsilon = 4\sigma_{\text{noise}}$.

energy consumption. For the remaining methods the complexity grows with $n^\star$. For PLAMLiS, this is due to the first step of the algorithm, where for each point the longest segment that meets the given error tolerance has to be found, see Section 2. When $x(n)$ is highly correlated, these segments become longer and PLAMLiS has to check a large number of times the tolerance constraint for each of the N samples of $x(n)$. For M-AAR and PR every time a new sample is added to a model (autoregressive

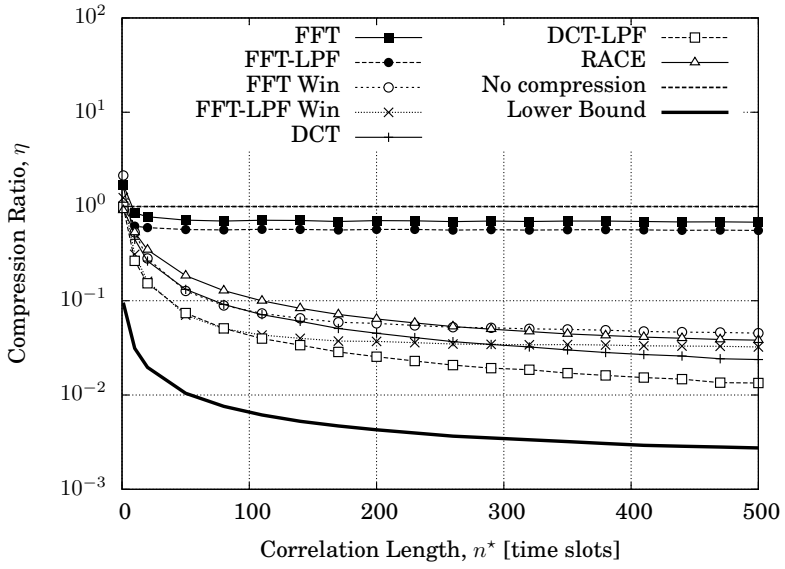
for the former and polynomial for the latter), this model must be updated and the error tolerance constraint has to be checked. These tasks have a complexity that grows with the square of the length of the current model. Increasing the correlation length of the input time series also increases the length of the models, leading to smaller compression ratios and, in turn, a higher energy consumption.

Fourier- and Wavelet-based Methods: we now analyze the performance of the Fourier- and Wavelet-based compression schemes of Section 2. We consider the same simulation setup as above. Fig. 4(a) shows that the compression performance of Fourier-based methods still improves with increasing $n^\star$. The methods that perform best are FFT Windowed, FFT-LPF Windowed and DCT-LPF, which achieve very small compression ratios, e.g., η is around 10^{-2} for $n^\star \geq 300$. Conversely, FFT and FFT-LPF, due to their edge discontinuity problem (see Section 2), need to encode more coefficients to meet the prescribed error tolerance constraint and thus their compression ratio is higher, i.e., around 1. RACE is outperformed by other DCT-based solutions in terms of compression performance, at all correlation lengths. As will be discussed shortly, this scheme may be interesting for its lightweight character in terms of energy consumption requirements. The energy cost for compression is reported in Fig. 4(b), where $n^\star$ is varied as an independent parameter. The compression cost for all the FFT/DCT schemes is given by a first contribution, which represents the energy needed to evaluate the FFT/DCT of the input signal $x(n)$. Thus, there is a second contribution which depends on the number of transformation coefficients that are picked. Specifically, a decreasing $n^\star$ means that the signal is less correlated and, in this case, more coefficients are to be considered to meet a given error tolerance. Further, for each of them, an inverse transform has to be evaluated to check whether an additional coefficient is required. This leads to an increasing computational cost for decreasing $n^\star$.

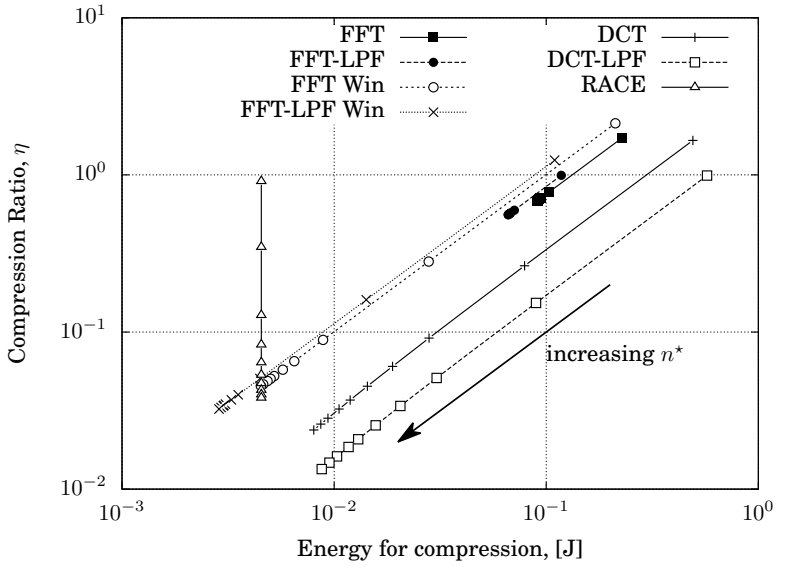
RACE instead, as described in Section 2.1.5, only performs an initial Wavelet decomposition and subsequently checks the reconstruction error thanks to the coefficient selection phase along the constructed tree, without having to compute an inverse transform at each step. Hence, its energy consumption remains nearly constant while varying the correlation length $n^\star$ and is lower than that of FFT and DCT schemes. Finally, we note that FFT-based methods achieve the best performance in terms of compression ratio among all schemes of Figs. 3(b) and 4(b) (DCT-LPF is the best performing algorithm), whereas PLA schemes give the best performance in terms of energy consumption for compression (LTC is the best among them).

Applicability to real-world signals: in Table I, we show the typical sampling rate and the correlation length for selected real-world signals. Luminosity and temperature data are taken from the database used in [Quer et al. 2012], readings from load sensors are taken from a structural monitoring WSN installed by WorldSensing in the Palau Sant Jordi of Barcelona (ES), whereas seismic data is obtained from the measurements in [Vilajosana et al. 2007]. The high quality (HQ) musical sample and the speech data are respectively from an excerpt of classical music by Mozart and from a sample of speech from an adult female, these datasets are available at [Donohue 2013]. The low quality (LQ) musical sample is from the Händel Messiah's Hallelujah Chorus.

In this paper, we focus on compression schemes for WSNs that are signal-agnostic and, as such, try to approximate the signals on the fly through some modeling technique. These, are however effective for slowly varying signals, say, with correlation length larger than 50 samples. Typically, these kinds of signals are monitored by WSNs gathering climatic/environmental data or structural health. Audio signals, such as music and voice, seismic signals, or signals related to online traffic monitoring show abrupt variations, are highly non-stationarity and are characterized by very short cor-



(a)



(b)

Fig. 4. (a) η *vs* Correlation Length $n^\star$ and (b) η *vs* Energy consumption for compression for the Fourier-based methods for fixed $\varepsilon = 4\sigma_{\text{noise}}$.

relation lengths (usually smaller than 10 samples). While the techniques presented here can be used for the compression of these signals, dedicated algorithms, which are outside the scope of this paper, are expected to lead to better results.

Table I. Typical correlation length $n^\star$ for selected real-world signals.

Signal type	Sampling rate [Hz]	Typical $n^\star$ [samples]
Indoor temperature	1/60	563
Humidity	1/600	355
Load sensors	1/5	402
Outdoor temperature	10	135
Luminosity	1/300	100
Music (HQ)	44.1 k	33
Music (LQ)	8192	4
Speech	8192	8
Seismic	150	3

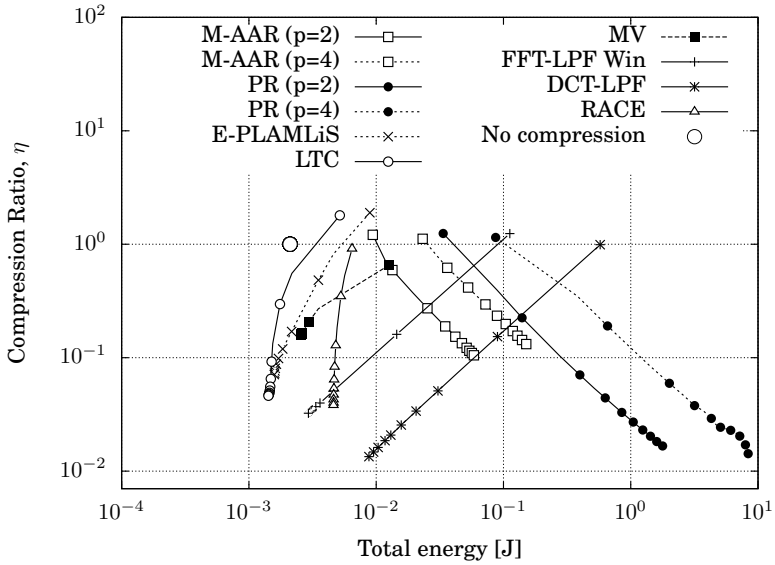


Fig. 5. Compression Ratio η *vs* Total Energy Consumption: comparison among lossy compression schemes.

3.7. Application Scenario

In this section, we evaluate the selected compression methods considering the energy consumed for transmission of typical radios in Wireless Sensor Networks (WSN) for a single- and a multi-hop network, where there is no multiple-user interference at the channel access. These results will be extended in Section 4 to Multi-hop networks with interference.

Single-hop Performance: Fig. 5 shows the performance in terms of Compression Ratio η *vs* Total Energy Consumption for a set of compression methods when applied to an interference-free single-hop WSN scenario. PLAMLiS was not considered as its performance is always dominated by E-PLAMLiS and we only show the performance of the best Fourier-based schemes. In both graphs the large white dot represent the case where no compression is applied to the signal, which is entirely sent to the gathering node. Note that energy savings can only be obtained for those cases where the total energy lies to the left of the no compression case. For the following results, we have set the transmission power of the radio transceiver to the maximum level, in order to show the best achievable performance when data compression is applied.

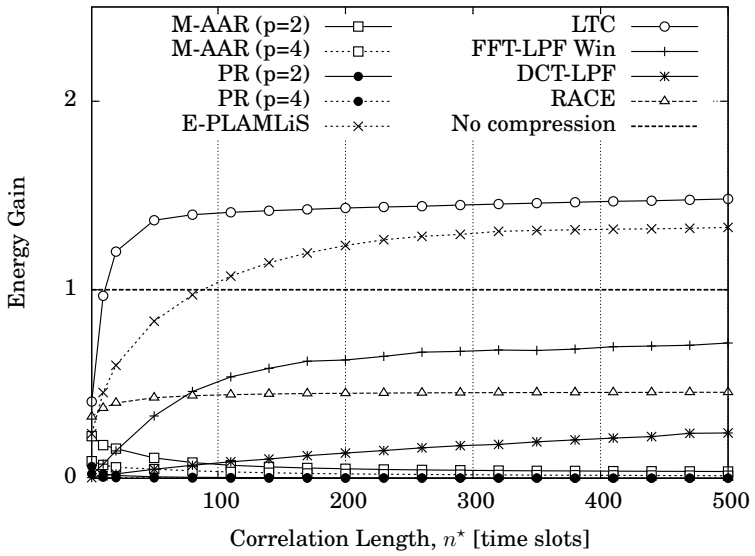


Fig. 6. Maximum Energy Gain *vs* Correlation Length $n^\star$ for a single-hop scenario.

Table II. Summary of performance for the considered compression methods.

Compression Method	Compression capabilities	Energy Requirements	Energy Efficiency	Complexity versus $n^\star$
PLAMLiS	average	high	×	increasing
E-PLAMLiS	average	low	✓	decreasing
LTC	average	low	✓	decreasing
PR	very high	very high	×	increasing
M-AAR	low	high	×	increasing
MV	low	moderate	×	decreasing
FFT	low	very high	×	decreasing
FFT-LPF	low	very high	×	decreasing
FFT Win	high	high	×	decreasing
FFT-LPF Win	very high	high	×	decreasing
DCT	high	high	×	decreasing
DCT-LPF	very high	high	×	decreasing
RACE	average/high	moderate	×	constant

Notably, in spite of the adoption of the maximum power level, the computational energy is comparable to that spent for transmission, thus, only LTC and Enhanced PLAMLiS can achieve some energy savings (see Fig. 5). All the other compression methods entail a high number of operations and, in turn, perform worse than the no compression case in terms of overall energy expenditure. This remarkable result is a consequence of that, as mentioned in Section 3.3, using current technologies, only a few hundred CPU instructions can be executed to compress a single bit of information and be energy efficient.

The total energy gain, defined as the ratio between the energy spent for transmission in the case with no compression and the total energy spent for compression and transmission using the selected compression techniques, is shown in Fig. 6. The method that offers the highest energy gain is LTC, although other methods such as DCT-LPF can achieve better compression performance (see Fig. 5). Note that in this scenario the

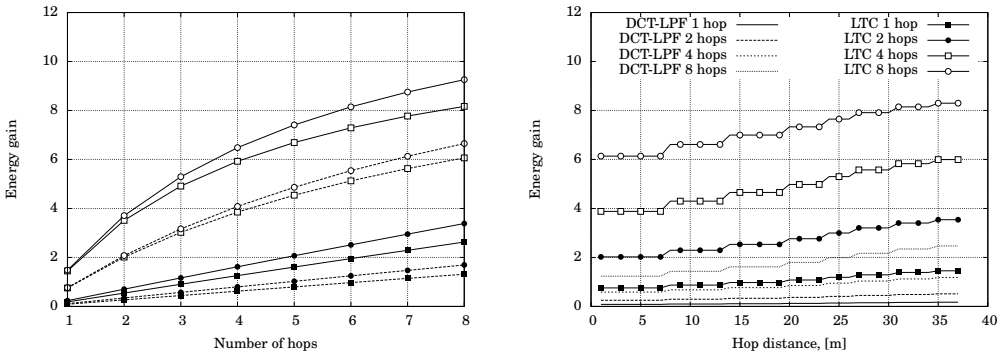


Fig. 7. (a) Energy gain vs number of hops for $\varepsilon = 4\sigma_{\rm noise}$. Solid lines are used to indicate maximum transmission power, dashed lines to indicate minimum transmission power. Results for DCT-LPF are shown with black filled markers, whereas white filled markers are used for LTC. The type of marker indicates the correlation length of the input signal, specifically: $(\square, \blacksquare)$ for $n^\star = 300$, $(\circ, \bullet)$ for $n^\star = 500$. (b) Energy gain vs hop distance for LTC and DCT-LPF with $\varepsilon = 4\sigma_{\rm noise}$ and $n^\star = 300$.

total energy is highly influenced by the computational cost. Thus, the most lightweight methods, such as LTC and enhanced PLAMLiS, perform best.

In Table II, we qualitatively summarize the performance of the considered signal compression algorithms, classifying them in terms of compression capabilities, energy requirements (directly related to their computational complexity) and dependence on the temporal correlation length $n^\star$.

Multi-hop Performance: in Fig. 7 we focus on multi-hop networks, and evaluate whether further gains are possible when the compressed information has to travel multiple hops to reach the data gathering point. In this case, both transmission and reception energy are accounted for at each intermediate relay node. In the following, only LTC and DCT-LPF are shown, as these are the two methods that respectively perform best in terms of complexity and compression efficiency.

In Fig. 7(a), we set the error tolerance $\varepsilon = 4\sigma_{noise}$, the correlation length of the input signal $n^\star \in \{300, 500\}$ and we evaluate the possible gains for the maximum and the minimum transmission power levels, so as to respectively obtain the upper and lower bounds on the achievable performance. As shown in this figure, the energy gain increases with the number of hops. This is because, although the energy spent for the compression at the source node is comparable to that spent for the transmission, the compression cost (compression energy) is only incurred at the source node; while each additional relay node only has to send compressed data. We also note that DCT-LPF is not energy efficient in single-hop scenarios, but it can actually provide some energy gains when the number of hops is large enough (e.g., larger than 2), and the transmission power is set to the maximum level. For the minimum transmission power, DCT-LPF starts being energy efficient only after $5 - 6$ hops, see Fig. 7(a).

In Fig. 7(b) we show the maximum achievable energy gain versus the distance between hops. Given the distance, the transmission power is selected according to the Friis path loss formula (with path loss exponent $\alpha = 3.5$, which is typical for WSNs [Mao et al. 2006]), considering the transmission power levels and the receiver sensitivity $P_{\rm th} = -95$ dBm of the CC2420 transceiver [Chipcon 2007]. For each value of the distance, we evaluated the energy gain using the minimum transmission power level that leads to a received power above P_{th}. As shown in Fig. 7(b), the energy gain

increases with the distance, as the transmission power becomes progressively higher of that needed for compression. This effect becomes more pronounced when the number of hops is increased, as the relay nodes only have to forward the data (no processing), thus benefiting from the smaller number of bits to be received and transmitted.

3.8. Numerical Fittings

In this section, we provide close-formulas to accurately relate the achievable compression ratio η to the relative error tolerance ξ and the computational complexity, N_c, which is expressed in terms of number of clock cycles per bit to compress the input signal $x(n)$. These fittings have been computed for the best compression methods, namely, LTC and DCT-LPF.

Note that, until now, we have been thinking of η as a performance measure which depends on the chosen error tolerance $\varepsilon = \xi\sigma_{noise}$. This amounts to considering ξ as an input parameter for the compression algorithm. In the following, we approximate the mathematical relationship between η and ξ, by conversely thinking of ξ as a function of η, which is now our input parameter. N_c can as well be expressed as a function of η.

We found these relationships through numerical fitting, running extensive simulations with synthetic signals. The relative error tolerance ξ can be related to the compression ratio η through the following formulas:

$$\xi(n^\star,\eta) = \begin{cases} \dfrac{p_1\eta^2 + p_2\eta + p_3}{\eta + q_1} & \text{LTC} \\ \dfrac{p_1\eta^4 + p_2\eta^3 + p_3\eta^2 + p_4\eta + p_5}{\eta + q_1} & \text{DCT-LPF}\,, \end{cases} \tag{2}$$

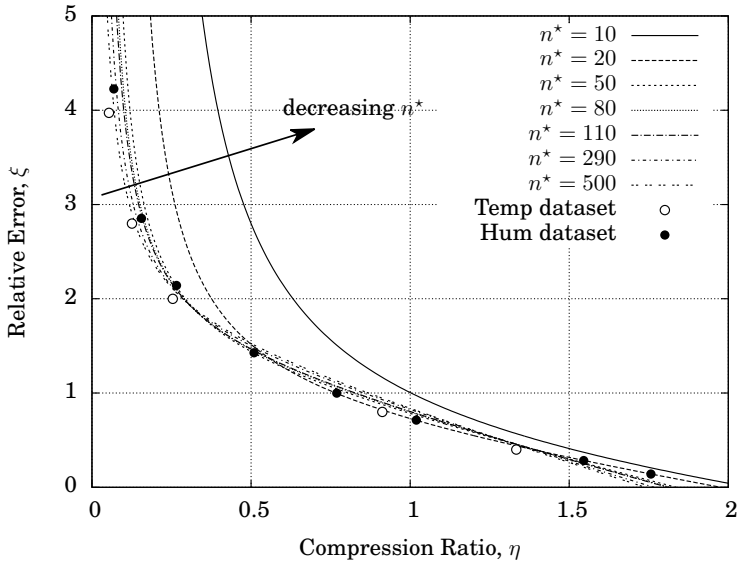
where the fitting parameters p_1, p_2, p_3, p_4, p_5, and q_1 depend on the correlation length $n^\star$ and are given in Table III for LTC and DCT-LPF. These fitting formulas have been validated against real world signals measured from the environmental monitoring WSN testbed deployed on the ground floor of the Department of Information Engineering (DEI), University of Padova, Italy [Crepaldi et al. 2007]. This dataset consists of measures of temperature and humidity, sensed with a sampling interval of 1 minute (temperature) and 10 minutes (humidity) for 6 days. Correlation lengths are $n^\star_T = 563$ and $n^\star_H = 355$ for temperature and humidity signals, respectively. The empirical relationships of Eq. (2) are shown in Fig. 8(a) and 8(b) through solid and dashed lines, whereas the markers indicate the performance obtained applying LTC and DCT-LPF to the considered real datasets. As can be noted from these plots, although the numerical fitting was obtained for synthetic signals, Eq. (2) closely represents the actual tradeoffs. Also, with decreasing $n^\star$ the curves relating ξ to η remain nearly unchanged in terms of functional shape but are shifted toward the right. Finally, we note that the dependence on $n^\star$ is particularly pronounced at small values of $n^\star$, whereas the curves tend to converge for increasing correlation length (larger than 110 in the figure).[2]

For the computational complexity, we found that N_c scales linearly with η for both LTC and DCT-LPF. Hence, N_c can be expressed through a polynomial as follows:

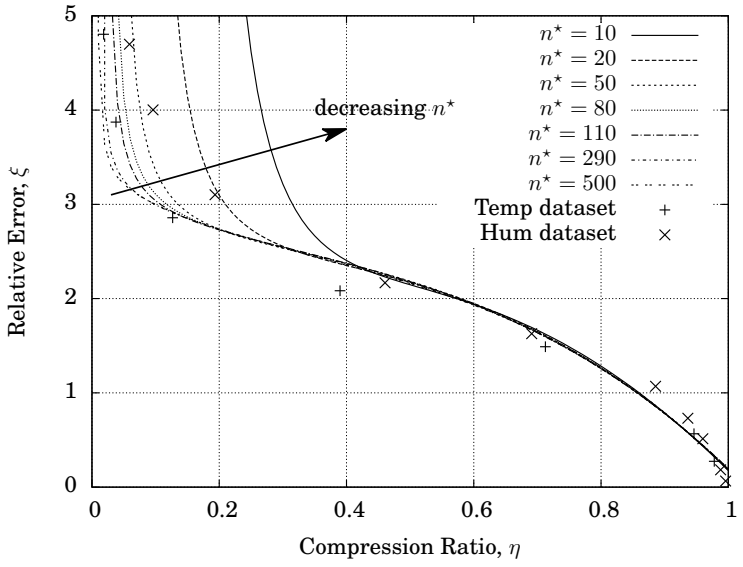
$$N_c(n^\star,\eta) = \alpha\eta + \gamma n^\star + \beta\,. \tag{3}$$

N_c exhibits a linear dependence on both $n^\star$ and η; the fitting coefficients are shown in Table IV. We remark that the computational complexity as given by (3) is that achievable by a temporal compressor configured with a compression ratio η, that keeps the reconstruction error bounded according to the error tolerance $\varepsilon = \xi(n^\star,\eta)\sigma_{\text{noise}}$ (see

[2]Note also that the there is a lower bound on the achievable reconstruction accuracy as the signal correlation $n^\star$ increases. This is due to the noise that is superimposed to the useful signal.



(a)



(b)

Fig. 8. Fitting functions $\xi(n^\star, \eta)$ *vs* experimental results: (a) LTC, (b) DCT-LPF. The correlation $n^\star$of the considered datasets for temperature and humidity is 563 and 355 samples, respectively.

(2)). Note that, differently from Fig. 4, this reasoning entails the compression of our data without fixing beforehand the error tolerance ε, which instead directly follows from η and $n^\star$.

Further, in (3) the dependence on $n^\star$ is much weaker than that on η and for practical purposes can be neglected without loss of accuracy. For this reason, in the remainder

Table III. Fitting coefficients for $\xi(n^\star, \eta)$.

Compression Method	$n^\star$	Fitting coefficients p_1	p_2	p_3	p_4	p_5	q_1
LTC	10	−0.35034	0.27640	0.92834	–	–	−0.15003
	20	−0.51980	0.86851	0.31368	–	–	−0.09245
	50	−0.80775	1.38842	0.17465	–	–	−0.03705
	80	−0.85691	1.45560	0.18208	–	–	−0.02366
	110	−0.86972	1.46892	0.19112	–	–	−0.01736
	290	−0.97242	1.61970	0.17280	–	–	−0.00747
	500	−1.03702	1.70305	0.17466	–	–	0.00267
DCT-LPF	10	2.05351	−12.70381	14.49624	−4.52198	0.82292	−0.16165
	20	−0.92752	−3.07506	3.07560	1.06902	0.02898	−0.09025
	50	−1.90344	−0.17491	−0.13500	2.43821	−0.03826	−0.03929
	80	−2.59629	1.41404	−1.40970	2.81971	−0.04122	−0.02667
	110	−2.57150	1.43655	−1.51646	2.87138	−0.02747	−0.01913
	290	−3.43806	3.17964	−2.67444	3.13226	−0.01531	−0.00848
	500	−3.99007	4.17811	−3.22636	3.22590	−0.01102	−0.00560

Table IV. Fitting coefficients for $N_c(n^\star, \eta)$.

Compression Method	Fitting coefficients α	β	γ
LTC	16.1	105.4	$3.1 \cdot 10^{-16}$
DCT-LPF	$48.1 \cdot 10^3$	82.3	$-2 \cdot 10^{-13}$

of this section we consider the simplified relationship:

$$N_c(\eta) = \alpha\eta + \beta \ . \tag{4}$$

The accuracy of Eq. (4) is verified in Fig. 9, where we plot our empirical approximations against the results obtained for the real world signals described above. The overall energy consumption is obtained as $N_b(x)N_c(\eta)E_0$.

Tradeoffs: in the following, we use the above empirical formulas to generalize our results to any processing and transmission technology, by separating out technology dependent and algorithm-dependent terms. Specifically, a compression method is energy efficient when the overall cost for compression ($E_c(x)$) and transmission of the compressed data ($E_{Tx}(\hat{x})$) is strictly smaller than the cost associated with transmitting $x(n)$ uncompressed ($E_{Tx}(x)$). Mathematically, $E_c(x) + E_{Tx}(\hat{x}) < E_{Tx}(x)$. Dividing both sides of this inequality by $E_{Tx}(x)$ and rearranging the terms leads to:

$$\frac{E_{Tx}(x)}{E_c(x)} = \frac{E'_{Tx}[\ell]N_b(x)}{E_0 N_c N_b(x)} > \frac{1}{1-\eta} \ ,$$

where the energy for transmission $E_{Tx}(x)$ is expressed as the product of the energy expenditure for the transmission of a bit $E'_{Tx}[\ell]$ (for the selected output power level $\ell \in \{1, \ldots, 8\}$) and the number of bits of $x(n)$, $N_b(x)$. The energy for compression is decomposed in the product of three terms: 1) the energy spent by the micro-controller in a clock cycle E_0, 2) the number of clock cycles performed by the compression algorithm per (uncompressed) bit of $x(n)$, N_c and 3) the number of bits composing the input signal $x(n)$, $N_b(x)$. With these energy costs and the above fitting Eq. (4) for N_c we can rewrite the above inequality so that the quantities that depend on the selected hardware architecture appear on the left hand side, leaving those that depend on algorithmic aspects

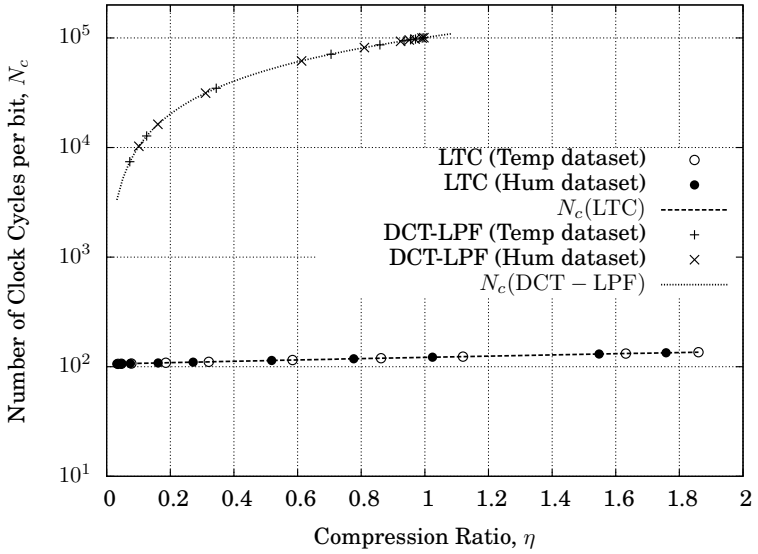


Fig. 9. Fitting functions $N_c(\eta)$ *vs* experimental results.

on the right hand side. The result is:

$$\frac{E'_{Tx}[\ell]}{E_0} > \frac{N_c(\eta)}{1-\eta} = \frac{\alpha\eta+\beta}{1-\eta} \ , \tag{5}$$

where α and β are the algorithmic dependent fitting parameters indicated in Table IV. Eq. (5) can be used to assess whether a compression scheme is suitable for a specific device architecture. As an example, for the considered WSN architecture we have that $E'_{Tx}[8] = 230$ nJ for the selected CC2420 radio for its highest transmission power , whereas for the TI MSP430 we have $E_0 = 0.726$ nJ and their ratio is $E'_{Tx}[8]/E_0 \simeq 316$. The numerical evaluation of the RHS of (5) for DCT-LPF reveals that this compression scheme is inefficient for any value of η, i.e., the overall energy expenditure due to transmission plus compression is higher than the energy spent in the case where compression is not applied. Instead, LTC provides energy savings for $\eta \leq 0.6$, that using the function $\xi(n^\star, \eta)$ for LTC can be translated into the corresponding (expected) error performance. Note that the knowledge of $n^\star$ is needed for this last evaluation. These results can be generalized to any other device technology, by comparing the RHS of (5) against the corresponding ratio $E'_{Tx}[\ell]/E_0$ and checking whether the inequality in (5) holds.

4. PERFORMANCE COMPARISON FOR INTERFERENCE-LIMITED MULTI-HOP NETWORKS

In this section we generalize our findings to multi-hop WSN where data is routed along a tree and eventually collected by a sink node. In doing so, we model the channel access dynamics in terms of transmission schedules, idle times and collisions, accounting for the corresponding energy and delay terms. For the sake of analytical tractability, we account for static routing paths. The main question that we try to answer here is whether additional benefits arise when further protocol inefficiencies are accounted for. Especially, we are concerned with the benefits that may be achieved for DCT schemes, which provide the best compression performance but that, as we have seen above, may be inefficient for single-hop networks when the channel access is idealized.

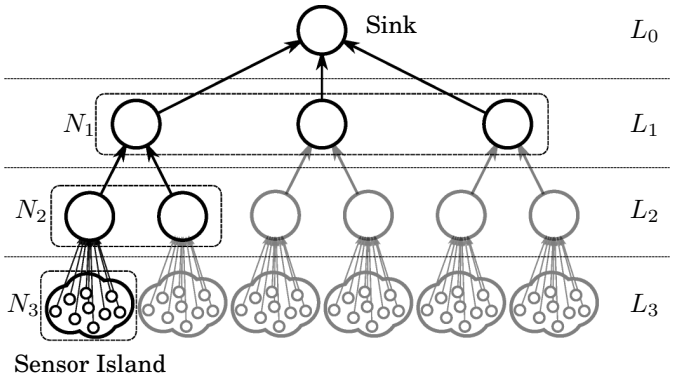


Fig. 10. Multi-hop WSN scenario.

4.1. Analysis for Interference Limited Networks

Scenario: we consider the multi-hop WSN of Fig. 10 where the field readings are gathered by the sensors placed in a number of WSN islands (at level L_{K-1}) and then routed to the data collector node (the WSN sink, at level L_0), through a data collection tree. This tree is organized according to a hierarchical structure with K levels, $L_0, L_1, \ldots, L_{K-1}$, whereby N_k is the number of children nodes for a root located at level $k-1$ with $k = 1, \ldots, K-1$ and $N_K = 0$. Thus, $N(k) = \prod_{i=1}^{k} N_i$, $k \geq 1$ is the total number of nodes at level k, $N(0) = 1$, and $\sum_{k=0}^{K-1} N(k)$ is the total number of sensor nodes, including the sink.

To reduce the interference among the data forwarding processes taking place in the different levels of the tree we adopt a pipelining scheduling technique as done in, e.g., [Rossi et al. 2010]. Starting with the lowest level L_{K-1}, the data collection protocol works in rounds of T seconds that are further subdivided into 3 sub-rounds, S_1, S_2 and S_3 of $T/3$ seconds each. During S_1, the $N(K-1)$ nodes at level L_{K-1} compete for the channel to send their data to their respective roots located in L_{K-2}, which act as receivers. During the next sub-round S_2, the nodes in L_{K-2} contend for the channel to forward their data to the nodes in the next level L_{K-3}, the nodes in L_{K-1} sleep and those in L_{K-3} act as receivers. In the final sub-round S_3, the nodes in L_{K-3} forward their data toward their upper level $K-4$ and those in levels $K-1$ and $K-2$ sleep. In the next sub-round (again of type S_1), the nodes at level L_{K-4} forward their data and those in L_{K-1} can concurrently transmit, being outside their interference range. This procedure is iteratively applied to each level so that the nodes that are three levels apart share the same schedule and concurrently transmit in the same sub-round. The nodes in each level $k = 1, 2, \ldots, K-2$ will receive in one sub-round, transmit in the next one, and stay silent in the last. The nodes in the last level L_{K-1} transmit in a sub-round and sleep during the following two. We further assume that the nodes belonging to the same level but to a different sub-tree do not interfere with each other.

The nodes in the sensor islands (L_{K-1}) are the only ones that generate endogenous traffic, each according to a Poisson process with rate λ_{K-1} pkt/s, which depends on their sampling rate for the underlying physical process. The rates for the nodes at the upper levels depend on the aggregated traffic, which is recursively computed accounting to MAC contentions and routing. The channel contentions during the transmission sub-rounds are governed through a protocol that is similar to S-MAC [Ye et al. 2002], extending the modeling approach of [Yang and Heinzelman 2012]. The authors of the

latter paper present a model for single-hop networks, where all nodes are allowed to talk to any other node. Below, we extend their analysis for the scenario of Fig. 10, where all the nodes in a certain level can only communicate with their respective root (located in the next level toward the sink). Hence, a mathematical analysis is developed to characterize the performance within each level, aggregating the contributions from L_{K-1} up to L_0 so as to obtain the overall network performance, i.e., from the WSN islands to the sink. Our extension is reported in what follows.

Analysis: we assume that each node has a finite FIFO queue that can store up to Q packets. During a transmission sub-round, only the nodes with non empty queue wake up and participate in the channel contention. In order to maintain synchronization across nodes, in S-MAC a fixed interval at the beginning of each active period is reserved for the exchange of SYNC packets. After this phase S-MAC exploits an RTS/CTS/DATA/ACK handshake to guarantee the successful transmission of the data. Specifically, each node access the channel after a backoff time picked uniformly at random from a fixed length contention window of $W\delta$ seconds, where W is the contention window size and δ is the time duration of an access slot. The first node that accesses the channel (the winner of the contention) sends an RTS packet and remains active until the completion of the RTS/CTS/DATA/ACK handshake, in which case its packet is successfully transmitted. All the remaining nodes go to sleep as soon as they overhear an RTS packet and wake up at the next transmission sub-round. If multiple nodes attempt to access the channel in the same access slot, their RTS packets will collide and the data packets of these nodes are dropped, causing a loss event. We also assume that the nodes remains active for the entire duration of their reception sub-rounds.

The analysis that follows is applied to each level of the discussed multi-hop WSN, starting from the sensor islands and recursively moving towards the sink. The MAC queue of each node in level $k = 0, 1, \ldots, K-1$ is modeled through a Markov chain, with transition probabilities depending on the corresponding arrival rate λ_k and on the probability $p \triangleq p_s + p_f$ of removing one packet from the queue (either due to a successful transmission, w.p. p_s, or due to a collided one, w.p. p_f).[3] For the levels above L_{K-1} we calculate λ_k aggregating the traffic of the N_{k+1} underlying nodes and accounting for the respective packet losses, as follows:

$$\lambda_k = \mathrm{PDR}_{k+1} \cdot N_{k+1} \cdot \lambda_{k+1} \qquad \forall\, k = 0, 1, \ldots, K-2\,, \tag{6}$$

where PDR_{k+1} is the average packet delivery ratio for the level immediately below. The packet generation rate for the nodes in the WSN islands (level L_{K-1}) is application dependent as is denoted by γ pkt/s. Accounting for the reduction in the number of packets sent due to the utilization of temporal compression algorithms, these nodes generate endogenous data traffic according to rate λ_{K-1}, where:

$$\lambda_{K-1} = \begin{cases} \gamma & \mathrm{no-compression} \\ \gamma\eta & \mathrm{compression}\,. \end{cases} \tag{7}$$

Following [Yang and Heinzelman 2012], given the Markov model for the MAC queues, for each node the stationary probability of empty queue π_o can be evaluated as a function of p, i.e., $\pi_o = f(p)$.[4] Also, given the number of nodes in the same group (i.e., sharing the same root node), the probability of winning a contention p can be

[3] We remark that p, p_s and p_f are to be calculated for each level k, according to the aggregated input traffic coming from the previous level $k-1$. Here, we omit their explicit dependence on k for the sake of readability.
[4] In what follows, in addition to p_s and p_f, the index k is also omitted from p and π_o for the sake of a more concise and readable notation.

expressed as a further function of π_o. i.e., $p = p_s + p_f = g(\pi_o)$. In fact, note that p_s and p_f depend on the number of nodes at level k that have at least one packet in their buffer and thus transmit toward the same root located at level $k-1$. This number is a r.v. having a binomial mdf with parameters N_k and π_0, i.e., $B(N_k, \pi_o)$. For each level $k = 1, \ldots, K-1$, the pair (p, π_o) is evaluated numerically by finding the only point where the two curves $f(p)$ and $g(\pi_o)$ intersect. This returns the (steady-state) operational point for the MAC system, which is successively used to evaluate the performance of interest, in particular: 1) the Packet Delivery Ratio, 2) the Delay and 3) the Mean Energy Consumption.

1) Packet Delivery Ratio (PDR): the PDR for a given level k, (PDR_k), is defined as the ratio of successfully delivered packets over the total number of generated data packets and derived as follows:

$$\mathrm{PDR}_k = \frac{(1-\pi_0)p_s}{\lambda_k T} \ .$$

For our multi-hop network, the Total Packet Delivery Ratio from level L_{K-1} up to the sink is obtained as $\prod_{k=1}^{K-1} \mathrm{PDR}_k$.

2) Delay: the delay for a given level k, D_k, is defined as the number of time slots needed to deliver a data packet to the next hop. The average delay for each level can be obtained through the analysis in [Yang and Heinzelman 2012]; the total average delay is obtained through the aggregation of the delays experienced in each level from the WSN islands all the way to the sink.

3) Mean Energy Consumption: for the energy consumption analysis, we recall the features of our version of S-MAC.

(1) The time slot is divided in three sub-rounds, all the nodes in the same level share the same schedule and are synchronized.
(2) In the transmission sub-round only the nodes with non empty queue wake up and contend for the channel.
(3) The nodes that wake up perform a SYNC phase, which is followed by a channel contention. Only the winner of this contention transmits a data packet, whereas all other nodes go to sleep as soon as they overhear an RTS packet or detect a collision.
(4) The winner of the contention goes to sleep as soon as the data packet is acknowledged (in the case of a successful transmission), or after an RTS collision (when two or more node access the channel in the same access slot).
(5) The nodes remain in listening(sleep) mode for the whole duration of their reception(sleep) sub-round.

With these assumptions, we define the Mean Energy per time slot E as the sum of the mean energy spent in the three sub-rounds:

$$E = E_{\mathrm{S}_1} + E_{\mathrm{S}_2} + E_{\mathrm{S}_3} \ . \tag{8}$$

Without loss of generality we hereby consider S_1 as the transmission sub-round, S_2 as the reception sub-round and S_3 as the sleep sub-round.

The average energy consumption in the transmission sub-round can be further factorized as:

$$E_{\mathrm{S}_1} = E_{\mathrm{sync}} + E_{\mathrm{data}} + E_{\mathrm{sleep}} \ , \tag{9}$$

where $E_{\rm sync}$, $E_{\rm data}$ and $E_{\rm sleep}$ respectively account for the average energy consumption in the SYNC, the data transmission and the sleep phase occurring within a sub-round of type S_1. Assuming that a the duration of a SYNC phase is $T_{\rm sync}$, that the transmission of a SYNC packet takes $t_{\rm SYNC}$ and that a node transmits a SYNC packet every $N_{\rm sync}$ time slots, $E_{\rm sync}$ is obtained as:

$$E_{\rm sync} = \frac{t_{\rm SYNC} P_{\rm Tx} + (T_{\rm sync} - t_{\rm SYNC}) P_{\rm Rx} + T_{\rm sync} P_{\rm Rx} (N_{\rm sync} - 1)}{N_{\rm sync}} \,, \tag{10}$$

where $P_{\rm Tx}$ and $P_{\rm Rx}$ are the radio power consumption for transmission and reception, respectively.

In order to evaluate $E_{\rm data}$ and $E_{\rm sleep}$, we consider a tagged node having a packet to send and look at the duration of its backoff window in the following three cases: C1) the node successfully transmits a data packet, C2) the RTS sent by this node collides, and C3) the node goes back to sleep as it detects channel activity before its backoff timer expires. The durations of these events is referred to as W_s, W_c and W_t for cases C1, C2 and C3, respectively (expressed in number of access slots). For a given number of nodes that take part in the contention, N_o, we obtain:

$$W_s(N_o) = \sum_{i=0}^{W-1} \frac{i \cdot \frac{N_o}{W} \cdot \left(\frac{W-i-1}{W}\right)^{N_o-1}}{\sum_{j=0}^{W-1} \frac{N_o}{W} \cdot \left(\frac{W-j-1}{W}\right)^{N_o-1}} \,, \tag{11}$$

$$W_c(N_o) = \sum_{i=0}^{W-1} \frac{i \cdot \left[\left(\frac{W-i}{W}\right)^{N_o} - \left(\frac{W-i-1}{W}\right)^{N_o} - \frac{N_o}{W} \cdot \left(\frac{W-i-1}{W}\right)^{N_o-1}\right]}{\sum_{j=0}^{W-1} \left(\frac{W-j}{W}\right)^{N_o} - \left(\frac{W-j-1}{W}\right)^{N_o} - \frac{N_o}{W} \cdot \left(\frac{W-j-1}{W}\right)^{N_o-1}} \,, \tag{12}$$

$$W_t(N_o) = \sum_{i=0}^{W-1} \frac{i \cdot \left[\left(\frac{W-i}{W}\right)^{N_o} - \left(\frac{W-i-1}{W}\right)^{N_o}\right]}{\sum_{j=0}^{W-1} \left(\frac{W-j}{W}\right)^{N_o} - \left(\frac{W-j-1}{W}\right)^{N_o}} \,. \tag{13}$$

Inside the sum of (11) we have the conditional probability that the tagged node wins the contention, accessing the channel in slot i, whereas all other nodes that participate in the contention pick an access slot greater than i, given that the contention is successful. The remaining equations (12) and (13) are obtained similarly, accounting for C2 and C3. The expected values for these durations are obtained averaging W_s, W_s and W_s over the possible values of N_o, as:

$$\mathrm{E}\left[W_{\{s,t,c\}}\right] = \sum_{n=1}^{N} \binom{N}{n} \cdot \pi_o^{N-n} \cdot (1-\pi_o)^n \cdot W_{\{s,t,c\}}(n) \,. \tag{14}$$

The mean durations in seconds is evaluated as $T_{\{s,c,t\}} = \mathrm{E}\left[W_{\{s,t,c\}}\right] \cdot \delta$.

Finally $E_{\rm data}$ and $E_{\rm sleep}$ can be computed accounting for cases C1 (w.p. $(1-\pi_o)\cdot p_s$), C2 (w.p. $(1-\pi_o)\cdot p_f$), C3 (w.p. $(1-\pi_o)\cdot(1-p_s-p_f)$) and a further case C4) where the tagged node does not participate in the contention as its queue is empty (w.p. π_o):

$$\begin{aligned} E_{\rm data} = {} & (1-\pi_o)\cdot p_s \cdot [(t_{\rm RTS} + t_{\rm DATA}) \cdot P_{\rm Tx} \\ & + (t_{\rm CTS} + t_{\rm ACK} + T_s) \cdot P_{\rm Rx} + E_c] \\ & + (1-\pi_o)\cdot p_f \cdot [t_{\rm RTS} \cdot P_{\rm Tx} + (t_{\rm CTS} + T_c) \cdot P_{\rm Rx} + E_c] \\ & + (1-\pi_o)\cdot(1-p_s-p_f) \cdot [(t_{\rm RTS} + T_t) \cdot P_{\rm Rx}] \\ & + \pi_o \cdot 0 \,, \end{aligned} \tag{15}$$

$$
\begin{aligned}
E_{\text{sleep}} = (1-\pi_o)\cdot p_s \cdot (T/3 - T_{\text{sync}} - t_{\text{RTS}} - t_{\text{CTS}} \\
- t_{\text{DATA}} - t_{\text{ACK}} - T_s)\cdot P_{\text{Sl}} \\
+ (1-\pi_o)\cdot p_f \cdot (T/3 - T_{\text{sync}} - t_{\text{RTS}} - t_{\text{CTS}} - T_c)\cdot P_{\text{Sl}} \\
+ (1-\pi_o)\cdot (1-p_s-p_f)\cdot (T/3 - T_{\text{sync}} - t_{\text{RTS}} \\
- T_t)\cdot P_{\text{Sl}} + \pi_o \cdot (T/3)\cdot P_{\text{Sl}}\ . \qquad (16)
\end{aligned}
$$

Here, t_{RTS}, t_{DATA}, t_{CTS}, t_{ACK} represent the time for transmitting an RTS, DATA, CTS and an ACK, respectively. P_{Sl} is the power consumption of the radio transceiver in the sleep mode. E_c in (15) is the energy consumption for the compression of a data packet, evaluated according to the packet length, the required number of CPU cycles (see (4)) and the energy consumption per CPU cycle E_0 ($E_c = 0$ if data is sent uncompressed). The probabilities p_s, p_f and π_o are computed as in [Yang and Heinzelman 2012].

4.2. Results

In this section we provide some results for the scenario of Fig. 10, where we consider a network with $K = 4$ levels, and with $N_1 = 3$, $N_2 = 2$ and $N_3 = 30$, leading to a total number of 190 nodes. The packet generation rate at the lowest level λ_3 is obtained considering an average inter-sampling time for the underlying physical phenomenon $I_s \in [10^{-3}, 10^2]$ seconds. Moreover, considering a packet payload size of 127 bytes and 16 bits per sample, the resulting packet generation rate for the endogenous traffic is $\gamma = 1/(63.5 I_s)$ packets/s. For the compression methods, we focus on LTC and DCT-LPF as from Section 3.8 we know that these respectively perform best in terms of computational complexity and compression efficiency. Thus, we compare their performance at different compression ratios, $\eta \in \{0.01, 0.1, 0.3, 0.5, 0.7, 0.9\}$, and without compression. The queue length of each node is set to $Q = 10$ pkts, its contention window to $W = 50$ access slots and the duration of an access slot is set to $\delta = 0.2$ ms. For the computation of the energy terms (compression and radio activity) we consider the power consumptions of the CC2420 radio transceiver and of the MSP430 micro-controller. In the results that follows, different performance metrics are shown as a function of the compression ratio η. As per our working methodology in this paper, each value of η (and, in turn, each curve in the following graphs) is characterized by a corresponding error tolerance $\xi(n^\star, \eta)$, attainable from the correlation length $n^\star$ and (2). A decreasing η corresponds to an increasing error as shown in Fig. 8.

In Figs. 11(a) 11(b) we plot the the total delivery delay as a function of the packet generation rate for each node in the WSN islands, γ. We consider the total delay incurred in compressing the data and transmitting them through multiple hops, as per the scenario of Fig. 10. For LTC, the delay is slightly longer than no compression at low data traffic (roughly $\gamma \leq 0.02$ packets/second), where the latter outperforms LTC by about 1.5 seconds (due to the additional execution time for compression). As γ gets larger, the delay performance when sending the data uncompressed is substantially impacted and this is due to the increased level of congestion in the network (i.e., number of collisions) and, in turn, to the longer waiting time that the packets experience in the network queues. The same fact occurs for LTC, where however the level of congestion is milder due to the lower amount of data that is injected into the network when compression is applied. Notably, LTC remains quite lightweight for all values of η and, thus, for all the corresponding values of ξ. This is because LTC requires a small number of operations (in fact, with LTC a single pass on the input signal is required to provide a compressed sequence meeting the required reconstruction fidelity ξ). DCT-LPF shows a totally different behavior due to our implementation of this scheme, see Section 2.1.2, whereby we iteratively check the reconstruction quality at the source. In general, one may use a different approach, by performing a single DCT transform for

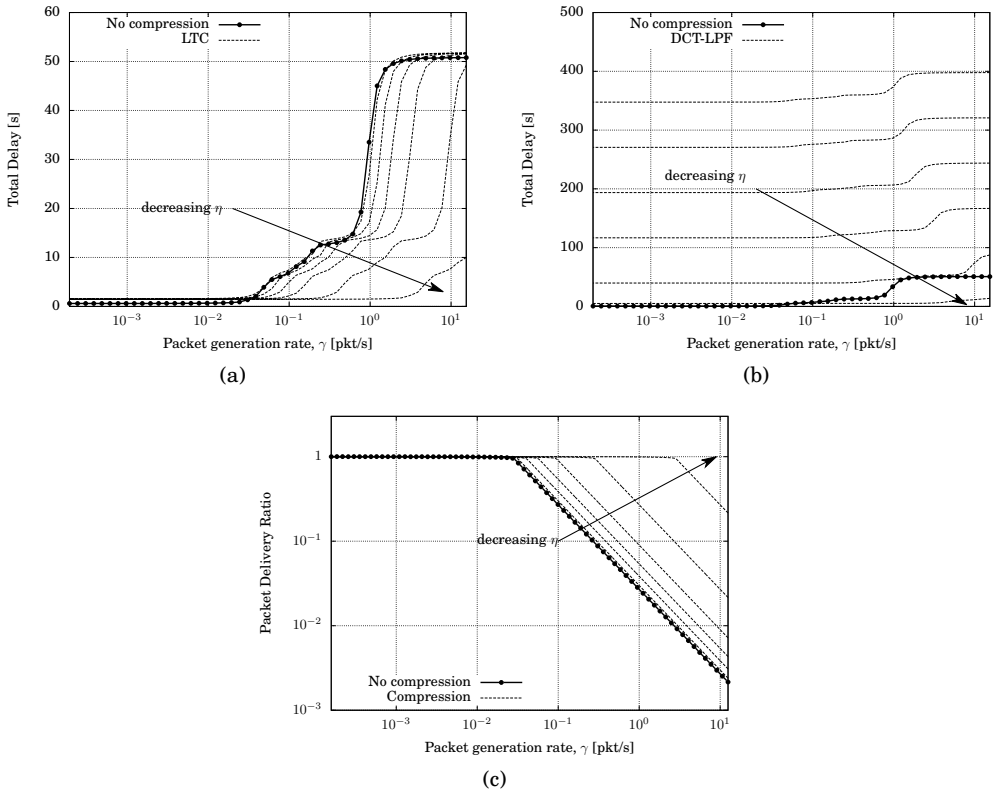


Fig. 11. Total Delay [s] *vs* packet generation rate γ at the lowest level for: (a) LTC; (b) DCT-LPF. (c) Packet Delivery Ratio *vs* packet generation rate γ at the lowest level.

each data frame and retaining a certain number of coefficients, so as to obtain to the wanted compression performance, irrespective of the resulting signal representation accuracy. In this case, the delay for DCT would only be slightly longer than that of LTC in Fig. 11(a). However, no control on the reconstruction quality can be assured in this case.

The Packet Delivery Ratio is shown in Fig. 11(c). When the arrival rate is small, all the packets are successfully delivered; conversely, as the arrival rate becomes larger, the queue lengths increase and the nodes start dropping packets. With temporal compression, the number of packets in the network is reduced and this leads to a higher PDR.

In Fig. 12, we investigate the energy balance arising from the tradeoff,compression versus transmission, by showing the overall average energy consumption over the entire network . Notably, for LTC, from Fig. 12(a) we see that the additional cost incurred in compressing the data is well counterbalanced by the higher efficiency of the channel access procedure due to the reduction in the data traffic, i.e., fewer packet collisions and more sleeping opportunities for the sensor nodes. However, this results does not hold when the compression method is DCT-LPF as in this case the cost associated with the compression of the data is much higher and some energy gain can only be achieved if the compression ratio is significant. Given these results, we advocate the use of linear compression methods such as LTC, as these are likely to lead to energy savings.

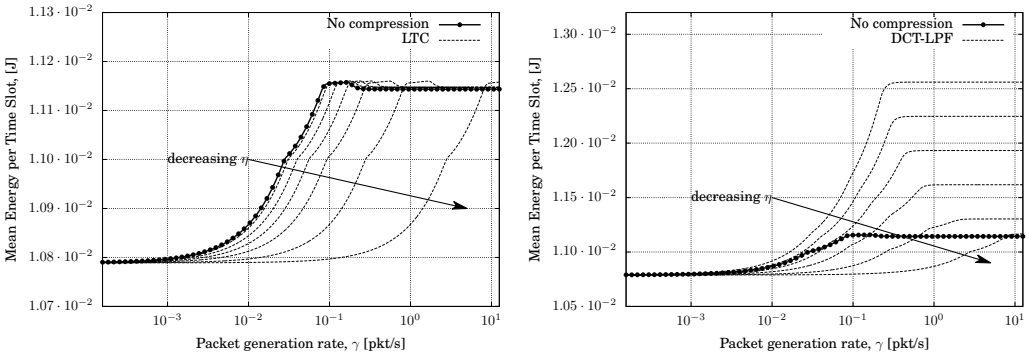


Fig. 12. Mean energy per time slot [J] *vs* packet generation rate γ at the lowest level: (a) LTC; (b) DCT-LPF.

The use of other methods should be carefully evaluated, as they are energy-inefficient in most practical cases and can lead to unacceptable delays .

Finally, in Fig. 13 we focus on the reconstruction performance for the network scenario of Fig. 10 in the presence of packet losses due to collisions and without accounting for any packet retransmission mechanism. To this end, we define a new reconstruction error metric, termed *frame Root Mean Square Error (RMSE)*, which is evaluated at the receiver as the RMSE for every N data samples. Since in both the compressed and uncompressed case the encoded data (corresponding to a window of N input samples) can be fragmented into multiple packets, we devised a simple loss tolerant decoding scheme for each method. In particular, 1) in the uncompressed case, whenever a packet is lost, the decoder uses the last valid received sample as its replacement. 2) For LTC, we assume that fragmentation only occurs between subsequent segment extremes $(n_i, x(n_i))$ (see Fig. 2) but that, for each of them, no splitting of n_i and $x(n_i)$ occurs across different packets. In this way, the receiver can always reconstruct the segments starting from a valid set of points (although in the case of a packet loss a longer segment will be used to represent the lost data points). 3) For DCT, the lost coefficients are considered equal to zero when the inverse transform is applied at the receiver.

As shown in Fig. 13, when the packet generation rate γ is small ($\gamma < 10^{-2}$ in the figure), the frame RMSE is constant and only depends on η and on the selected compression scheme. In this case, LTC shows a higher average RMSE than DCT-LPF for the same compression ratio η, as seen in Section 3.8. As γ increases beyond a certain threshold $\gamma_{\rm th}$, the transmission channel gets saturated and, in turn, packets start being lost due to collisions. Note that this occurs earlier for the uncompressed case and for those cases where the compression does not effectively reduce the amount of traffic in the network (e.g., $\eta = 0.9$). This leads to an abrupt increase of the frame RMSE as γ increases beyond the channel saturation point $\gamma_{\rm th}$.

On the other hand, when the compression schemes are configured so as to effectively reduce the amount of data that is transmitted over the network ($\eta > 0.5$ in Fig. 13), we have that $\gamma_{\rm th}$ moves to the right and this produces a beneficial effect in terms of frame RMSE, which remains constant and acceptable for a wider range of transmission rates. As an example, when the compression ratio is $\eta = 0.1$, the signal reconstruction error at the receiver can be kept small up to a packet generation rate that is roughly one order of magnitude larger than $\gamma_{\rm th}$ in the uncompressed case.

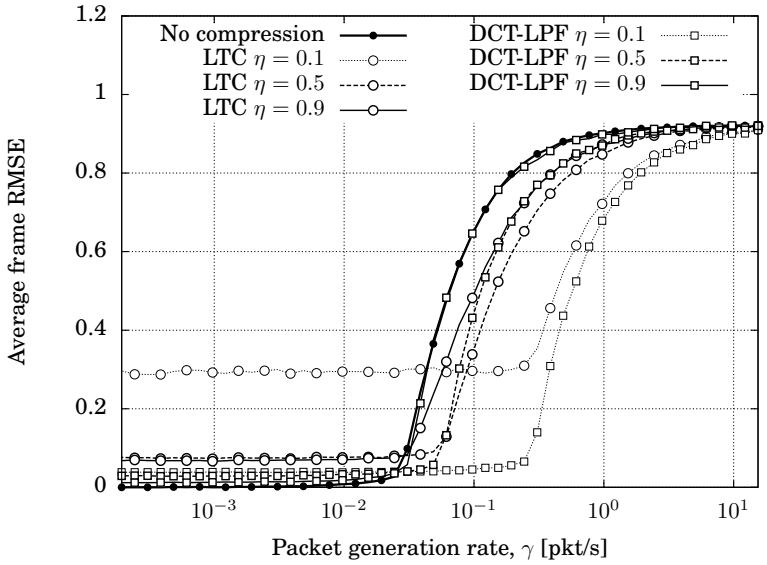


Fig. 13. Average frame RMSE *vs* packet generation rate γ for an input signal with $n^\star = 290$.

5. CONCLUSIONS

In this paper we have systematically compared lossy compression algorithms for constrained sensor networking, by investigating whether energy savings are possible depending on signal statistics, compression performance and hardware characteristics. Our results reveal that, for wireless transmission scenarios, the energy required by compression algorithms has the same order of magnitude of that spent for transmission at the physical layer. In this case, the only class of algorithms that provides some energy savings is that based on piecewise linear approximations, as these algorithms have the smallest computational cost. We have additionally obtained fitting formulas for the best compression methods to relate their computational complexity, approximation accuracy and compression ratio performance. These have been validated against real datasets and can be used to assess the effectiveness of the selected compression schemes for further hardware architectures. In the last part of the paper we investigate how these compression schemes perform in terms of energy efficiency, reduced network delay and increased reliability for multi-hop networks in the presence of realistic channel access procedures. Interestingly, we find that linear compression (e.g., LTC) is beneficial in all cases but more energy-hungry DCT methods often perform worse than no compression in terms of energy expenditure. The use of the latter is thus discouraged and should be carefully evaluated depending on the specific scenario at hand. Also, when packet losses affect the data delivery, as due to, e.g., packet collisions, correctly configured compression schemes also outperform the uncompressed transmission case in terms of reconstruction fidelity at the receiver. Essentially, this is due to the corresponding reduction in the transmitted data traffic and, in turn, in the packet collision probability.

REFERENCES

ABRAHAMSEN, P. 1997. A Review of Gaussian Random Fields and Correlation Functions. Tech. rep., Norwegian Computing Center, Box 114, Blindern, N-0314 Oslo, Norway.

ATZORIA, L., IERA, A., AND MORABITO, G. 2010. The Internet of Things: A survey. *Computer Networks 54,* 15, 2787–2805.

BARR, K. C. AND ASANOVIĆ, K. 2006. Energy-aware lossless data compression. *ACM Transactions on Computer Systems 24,* 3, 250–291.

BENZI, F., ANGLANI, N., BASSI, E., AND FROSINI, L. 2011. Electricity Smart Meters Interfacing the Households. *IEEE Transactions on Industrial Electronics 58,* 10, 4487–4494.

BERGER, T. 1971. *Rate Distortion Theory: A Mathematical Basis for Data Compression*. Prentice Hall.

BIERL, L. 2000. MSP430 Family Mixed-Signal Microcontroller Application Reports. Tech. rep., Texas Instruments Incorporated.

CHEN, H., LI, J., AND MOHAPATRA, P. 2004. RACE: time series compression with rate adaptivity and error bound for sensor networks. In *IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*. Fort Lauderdale, FL, US.

CHIPCON. 2007. SmartRF CC2420: 2.4 GHz IEEEE 802.15.4/ZigBee-ready RF Transceiver. Tech. rep., Texas Instruments Incorporated.

CREPALDI, R., FRISO, S., HARRIS III, A. F., MASTROGIOVANNI, M., PETRIOLI, C., ROSSI, M., ZANELLA, A., AND ZORZI, M. 2007. The Design, Deployment, and Analysis of SignetLab: A Sensor Network Testbed and Interactive Management Tool. In *IEEE Tridentcom*. Orlando, FL, US.

DAVIES, R. B. AND HARTE, D. S. 1987. Tests for Hurst effect. *Biometrika 74,* 1, 95–101.

DODSON, S. 2003. The Internet of Things. *The Guardian*.

DONOHO, D., VETTERLI, M., DEVORE, R., AND DAUBECHIES, I. 1998. Data compression and harmonic analysis. *Information Theory, IEEE Transactions on 44,* 6, 2435–2476.

DONOHUE, K. D. 2013. Wav data files for class examples and laboratory experiments. http://www.engr.uky.edu/~donohue/ee422/Data/DataEE422.htm.

FASOLO, E., ROSSI, M., WIDMER, J., AND ZORZI, M. 2007. In-Network Aggregation Techniques for Wireless Sensor Networks: A Survey. *IEEE Wireless Communication Magazine 14,* 2, 70–87.

IEEE P802.15 WORKING GROUP. 2003. Std. IEEE 802.15.4-2003: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). In *IEEE Standard*. 1–89.

KAPPLER, C. AND RIEGEL, G. 2004. A Real-World, Simple Wireless Sensor Network for Monitoring Electrical Energy Consumption. In *European Workshop on Wireless Sensor Networks (EWSN)*. Berlin, Germany.

LIANG, Y. 2011. Efficient temporal compression in wireless sensor networks. In *International Conference on Local Computer Networks (LCN)*. Bonn, Germany.

LIU, C., WU, K., AND PEI, J. 2007. An energy-efficient data collection framework for wireless sensor networks by exploiting spatio-temporal correlation. *IEEE Transactions on Parallel and Distributed Systems 18,* 7, 1010–1023.

LU, J.-L., VALOIS, F., AND DOHLER, M. 2010. Optimized Data Aggregation in WSNs Using Adaptive ARMA. In *International Conference on Sensor Technologies and Applications (SENSORCOMM)*. Venice, Italy.

MAO, G., ANDERSON, B. D. O., AND FIDAN, B. 2006. Online Calibration of Path Loss Exponent in Wireless Sensor Networks. In *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*. San Francisco, CA, US.

MARCELLONI, F. AND VECCHIO, M. 2009. An Efficient Lossless Compression Algorithm for Tiny Nodes of Monitoring Wireless Sensor Networks. *The Computer Journal 52,* 8, 969–987.

MARCELLONI, F. AND VECCHIO, M. 2010. Enabling energy-efficient and lossy-aware data compression in wireless sensor networks by multi-objective evolutionary optimization. *Elsevier Information Sciences 180,* 10, 1924–1941.

PALATTELLA, M., ACCETTURA, N., VILAJOSANA, X., WATTEYNE, T., GRIECO, L., BOGGIA, G., AND DOHLER, M. 2012. Standardized Protocol Stack for the Internet of (Important) Things. *IEEE Communications Surveys Tutorials, to appear PP,* 99, 1–18.

PATTEM, S. AND KRISHNAMACHARI, B. 2004. The impact of spatial correlation on routing with compression in wireless sensor networks. In *ACM/IEEE International Symposium on Information Processing in Sensor Networks (IPSN)*. Berkeley, CA, US.

PHAM, N. D., LE, T. D., AND CHOO, H. 2008. Enhance exploring temporal correlation for data collection in WSNs. In *IEEE International Conference on Research, Innovation and Vision for the Future (RIVF)*. Ho Chi Minh City, Vietnam.

PHILLIPS, G. M. 2003. *Interpolation and Approximation by Polynomials*. Springer.

QUER, G., MASIERO, R., PILLONETTO, G., ROSSI, M., AND ZORZI, M. 2012. Sensing, Compression and Recovery for WSNs: Sparse Signal Modeling and Monitoring Framework. *IEEE Transactions on Wireless Communications 11,* 10, 3447–3461.

RAO, K. R. AND YIP, P. 1990. *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Academic Press Professional, Inc., San Diego, CA, US.

ROSSI, M., BUI, N., ZANCA, G., STABELLINI, L., CREPALDI, R., AND ZORZI, M. 2010. SYNAPSE++: Code Dissemination in Wireless Sensor Networks using Fountain Codes. *IEEE Transactions on Mobile Computing 9,* 12, 1749–1765.

SADLER, C. M. AND MARTONOSI, M. 2006. Data compression algorithms for energy-constrained devices in delay tolerant networks. In *International Conference on Embedded Networked Sensor Systems (SenSys)*. Boulder, CO, US.

SCHOELLHAMMER, T., GREENSTEIN, B., OSTERWEIL, E., WIMBROW, M., AND ESTRIN, D. 2004. Lightweight temporal compression of microclimate datasets. In *IEEE International Conference on Local Computer Networks (LCN)*. Tampa, FL, US.

SHARAF, M., BEAVER, J., LABRINIDIS, A., LABRINIDIS, R., AND CHRYSANTHIS, P. 2003. TiNA: A Scheme for Temporal Coherency-Aware in-Network Aggregation. In *ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE)*. San Diego, CA, US.

SRISOOKSAI, T., KEAMARUNGSI, K., LAMSRICHAN, P., AND ARAKI, K. 2012. Practical data compression in wireless sensor networks: A survey. *Elsevier Journal of Network and Computer Applications 35,* 1, 37–59.

SZEWCZYK, R., OSTERWEIL, E., POLASTRE, J., HAMILTON, M., MAINWARING, A., AND ESTRIN, D. 2004. Habitat monitoring with sensor networks. *ACM Communications 47,* 6, 34–40.

VAN DER BYL, A., NEILSON, R., AND WILKINSON, R. 2009. An evaluation of compression techniques for Wireless Sensor Networks. In *AFRICON*. Nairobi, Kenya.

VILAJOSANA, I., KHAZARADZE, G., SURIÑACH, E., LIED, E., AND KRISTENSEN, K. 2007. Snow avalanche speed determination using seismic methods. *Elsevier Cold Regions Science and Technology 49,* 1, 2–10.

VULLERS, R., SCHAIJK, R., VISSER, H., PENDERS, J., AND HOOF, C. 2010. Energy Harvesting for Autonomous Wireless Sensor Networks. *IEEE Solid-State Circuits Magazine 2,* 2, 29–38.

WALLACE, G. K. 1992. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics 38,* 1, xviii –xxxiv.

WERNER-ALLEN, G., LORINCZ, K., RUIZ, M., MARCILLO, O., JOHNSON, J., LEES, J., AND WELSH, M. 2006. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing 10,* 2, 18–25.

XU, N., RANGWALA, S., CHINTALAPUDI, K., GANESAN, D., BROAD, A., GOVINDAN, R., AND ESTRIN, D. 2004. A Wireless Sensor Network for Structural Monitoring. In *ACM Conference on Embedded Networked Sensor Systems (SENSYS)*. Baltimore, MD, US.

YANG, O. AND HEINZELMAN, W. 2012. Modeling and Performance Analysis for Duty-Cycled MAC Protocols with Applications to S-MAC and X-MAC. *IEEE Transactions on Mobile Computing 11,* 6, 905–921.

YE, W., HEIDEMANN, J., AND ESTRIN, D. 2002. An energy-efficient MAC protocol for Wireless Sensor Networks. In *IEEE INFOCOM*. Vol. 3. 1567–1576.

ZORDAN, D., MARTINEZ, B., VILAJOSANA, I., AND ROSSI, M. 2012. To Compress or Not To Compress: Processing vs Transmission Tradeoffs for Energy Constrained Sensor Networking. Tech. rep., Department of Information Engineering, University of Padova, Padova, Italy.

ZORDAN, D., QUER, G., ZORZI, M., AND ROSSI, M. 2011. Modeling and Generation of Space-Time Correlated Signals for Sensor Network Fields. In *IEEE Global Telecommunications Conference (GLOBECOM)*. Houston, TX, US.