2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS) | 978-1-6654-4935-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/MASS52906.2021.00019

Entropic Sensing for Energy Efficiency

M. Adel Ibrahim
Department of Electrical and Computer Engineering
Queen's University
Kingston ON, Canada
https://orcid.org/0000-0002-6173-410X

Hossam S. Hassanein
School of Computing
Queen's University
Kingston ON, Canada
https://orcid.org/0000-0003-0260-8979

Abstract—We present a novel energy-efficient approach to wireless real-time sensing. For a sensor node (SN) transmitting samples of a discrete time series in real-time, its lifetime depends largely on its battery capacity. With most of the energy consumed in wireless transmission, we present an energy efficient scheme that can significantly reduce the number of transmitted samples, while maintaining a low mean absolute error between the original and the recovered signals. We introduce the concept of instantaneous entropy and we derive a computationally efficient iterative formula for computing Shannon's entropy. The SN evaluates the information content in each sample and decide whether to transmit or omit the sample. At the sink, we use incremental machine learning to recover the omitted samples in real-time. Our approach showed an average of 60% reduction in energy consumption by the SN with less than 2% mean absolute error in the recovered signal.

Index Terms—Entropy, wireless sensing, sensor node, wireless sensor network, industrial internet of things, energy efficiency, time series prediction, incremental machine learning.

I. Introduction

The use of wireless sensor networks (WSNs) to monitor all kinds of industrial and environmental assets is on the rise. The ecosystem and the communication standards needed for the full adoption of WSNs are arguably mature enough. On the contrary, one crucial component in any WSN will always be hard to standardize that is the Sensor Node (SN) which is usually battery powered.

Every application has its own requirements which springs from a unique set of operating conditions, performance goals, and design constraints. Since application requirements are the main driver of hardware design decisions, therefore each application would need its own custom designed SN. The need for a different SN in each application is asserted by Healy, Newe, and Lewis in [1] after a thorough review of the hardware specifications for over 40 different SNs.

While hardware design requirements/challenges vary from one application to another, there is one pressing challenge that sprigs up in almost every design which is the limited energy budget of wireless sensors. Even with the recent advances in energy storage/harvesting technology, it is still one of the highly restricting factors in any wireless sensing hardware design.

Energy budgets can be balanced by either increasing supply or decreasing demand. Increasing energy supply relates directly to the capacity of the battery or the output power of a given harvesting technology. Both battery capacity and harvesting technologies are highly dependant on the application at hand. Thus, it is relatively harder to address the energy challenge from the supply side in a general sense. However, from the demand side, there is something to be done that can generally reduce power consumption in potentially any SN. In this paper, we demonstrate a method to reduce the power consumption of an SN by reducing the number of data packets an SN has to transmit in real-time.

An SN is usually composed of four major functional units: the power supply, the controller, the sensor/transducer, and the wireless transceiver. According to several works [2]–[5], the transceiver is by far the most power demanding unit in the SN. Thus, by reducing the number of instances of wireless transmissions, we can significantly decrease the demand for power on an SN.

In the context of monitoring applications where the WSN is deployed to closely monitor the condition/status of an industrial asset, the data is transmitted in real-time. In other words, the SN takes a measurement and transmits the reading back to the sink instantaneously. Our general approach is to maximize information transfer from an SN to its sink while minimizing data transfer. Each other approach to this problem puts forward its own assumptions and has its own limitations. We will briefly discuss two well-known approaches and explain the approach followed in this work at the end of this section.

Data compression aims to reduce the payload of each data packet before transmission [6], this can be an effective method when the data payload is relatively large such as when an SN is transmitting numerous sensor readings in a single packet.

On the other hand, the energy savings from reducing the payload can be limited even with a 67% compression ratio as claimed in [6] because, the overhead required by channel coding and network protocols for framing the data packet depreciates the power savings gained from reducing the size of the payload. Also, regardless of the payload size, the very act of activating the transceiver is power consuming.

978-1-6654-4935-9/21/$31.00 ©2021 IEEE
DOI 10.1109/MASS52906.2021.00019

Furthermore, in data compression algorithms, prior knowledge about the statistical distribution of the signal is assumed. This assumption is essential for designing the optimal code for the data (e.g., [6]). Thus, for different sensors, the SN will have different codes implemented to compress their readings before transmission. Since most physical sensory data are non-stationary, prior knowledge are inconclusive and subject to obsolescence.

Another approach is compressed sensing [7], [8]. Compressed sensing is built on convex optimization which assumes the availability of data in batches. Contrarily, in our case the data arrives as individual samples in a one-dimensional time series. While there has been some work on dynamic compressed sensing [9]–[11], the sparsity assumption is still essential in these works. In other words, the measurements are taken incoherently from a signal whose representation in a given basis transformation contains a sparse set of coefficients.

Compressed sensing is not applicable here for two reasons. First, sensory data in real-time monitoring applications do not arrive in batches. Second, the sparsity assumption cannot be made for any signal without prior knowledge about its structure.

Generally, in a WSN, the sink has significantly more energy and computing resources than an SN. Thus, our approach will build upon this difference in levels of energy supply and computing resources between the sink and an SN. By efficiently omitting information at the SN we can reduce wireless traffic and by extension, the SN's energy consumption. Then, at the sink we can afford to spend more energy and utilize more computing resources to recover the omitted information in real-time.

We aim to reduce the number of samples to be transmitted by the SN without causing too much error in the recovered data. The question is, how can the SN select the samples to transmit and the ones to omit? Generally, samples are not equally valuable in information sense. Some sample are interesting (contain a lot of information) while others are boring (contain little information). The goal is to omit the boring samples which are relatively easier to be predicted by the sink.

In Section II, a formal description of the problem is introduced along with our assumptions. In section III we present the entropic filtering algorithm. Section IV explains the chosen recovery method. The system evaluation is presented and discussed in section V. Finally, we conclude in section VI.

II. Problem Formulation

At every time instant, the SN will have knowledge of the current sample in addition to some accumulated statistical quantities of the past samples. Then, the SN will try to evaluate each sample (as it is captured) and determine whether it should be transmitted or omitted based on the sample's information content compared to the accumulated information content of the time series.

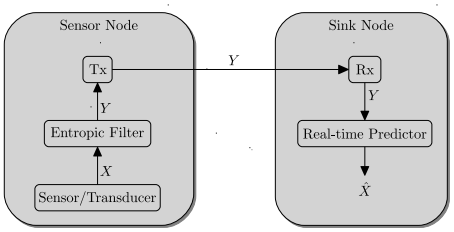

Fig. 1. The signal path in the introduced entropic sensing system.

Let us consider an SN monitoring a physical quantity (e.g., temperature, pressure, liquid flow, proximity) in real-time by regular sampling (i.e., samples are equally spaced in time). Samples are transmitted in real-time to a sink over a half-duplex channel. The sensory data stream forms a discrete time series X. Each sample is transmitted in real-time in a single data packet.

The intention is to reduce the energy consumption of the SN by reducing the number of transmitted samples while being able to recover the omitted samples at the sink node with tolerable error. This means the SN will cease to transmit a subset of X as they are collected in real-time.

As shown in Fig. 1, X is a discrete time series to be transmitted from an SN to a sink node. Y is the observed signal at the receiver which will be used to construct an estimate of X called $\hat{X}$. The solution then is to find the best estimate $\hat{X}$ for the current instance of X given the current and the past instances of Y and $\hat{X}$.

The series X has an alphabet $\mathcal{A}$ and a probability mass function (PMF) $P(x_n) = Pr\{X = x_n\}$ where $x_n \in \mathcal{A}$ such that $n \in \{1, 2, 3, ...N\}$ where $N = \#\mathcal{A}$, as in the cardinality of the set $\mathcal{A}$. In other words, N is the number of bins in the histogram of X. Furthermore, X produces uniform time samples represented by the vector $S = \{s_1, s_2, ..., s_t, ..., s_T\}$, where T is the total number of samples in X.

A. Assumptions

The following assumptions are made to set the framework of simulations and to outline the suitability of ES for different applications.

1) The probability of error in Y is zero. This assumption is made to simplify simulations. The point of this work is to demonstrate ES as an approach. Thus, introducing channel imperfections will not put the proposed methodology at any advantages/disadvantage over the existing methods because all are subject to the limitations of wireless channels. One may argue that ES reduces wireless traffic and therefore may be less prone to channel induced errors such as packet collisions.
2) X is a regularly sampled discrete time series. Meaning, time intervals between X's samples are equal.

Also, X's samples are quantized to a finite predefined set of bins.

B. Bench-marking time series

In this section we describe the set of discrete time series (signals) chosen to evaluate the performance of the proposed ES system. All signals were normalized to their maximum value before being processed. The normalization facilitates the comparison of performance between all signals, however, in a real application, no normalization is required.

1) A constant signal with a square pulse. This signal was introduced to make sure that samples representing significant unpredictable events in the signal will be picked up by the entropic filtering (EF) algorithm and transmitted in real-time.

$$S1 = \begin{cases} 1 & 0.5T \leq t \leq 0.6T \\ 0 & \text{Otherwise} \end{cases} \quad (1)$$

2) A linear monotonic signal increasing in time with a constant slope $S2 = 1.2t$. This signal evaluates the predictor's performance with a predictable input.
3) A simple sinusoidal signal with a single frequency and no added noise. This and the next three signals represent a sinusoidal waveform with a staggered increase in complexity. $S3 = A\sin(t)$, where $A = 10$ is the amplitude.
4) We add four equal-amplitude harmonics to the previous signal to get $S4$ without adding noise.

$$S4 = A\sin\left(\frac{t}{4}\right) + A\sin\left(\frac{t}{2}\right) + A\sin(t) + A\sin(5t) \quad (2)$$

5) To increase the complexity of $S5$, we add Additive White Gaussian Noise (AWGN) to $S4$. Thus, $S5 = S4 + A_n N(0, 0.8)$, where $A_n = 5$ is the noise amplitude.
6) We introduce an unpredictable event similar to $S1$ in order to evaluate the system's performance in picking up such events in real-time.

$$S6 = \begin{cases} S5 + 10A & 0.5T \leq t \leq 0.6T \\ S5 & \text{Otherwise} \end{cases} \quad (3)$$

7) $S7$ is a non-stationary signal from an iterative generator and AWGN as follows. Since, most physical signals are non-stationary, it will be interesting to see the system's performance in processing such signals.

$$s_t = s_{t-1} + A_n N(\mu = 0, \sigma = 0.8) \quad \forall\ t = \{2, 3, 4, ..., T\} \quad (4)$$

where, $s_1 = random(-1, 1)$ and $A_n = 20$, such that $random(x, y)$ is a random generator with a uniformly distributed binary output (x and y).

8) To push our system to its limits, we introduce a purely random signal. $S8$ is a random walk produced with an iterative generator as follows,

$$s_t = s_{t-1} + random(-1, 1) \quad \forall\ t = \{2, 3, 4, ..., T\} \quad (5)$$

where, $s_1 = random(-1, 1)$.
9) To conform with the current literature, we use $S9$ and $S11$ two of the most commonly used series in time series analysis and prediction research. $S9$ is the Santa Fe laser time series [12], [13].
10) $S10$ is called the Hénon map [14], a widely studied discrete-time dynamic system with chaotic output [13]. This is used to evaluate the system's performance in chaotic signals which are common in the physical world. $S10$ is generated as follows,

$$s_t = 1 - a{s_{t-1}}^2 + bs_{t-2} \quad \forall\ t = \{3, 4, ..., T\} \quad (6)$$

where $s_1 = s_2 = 0$, $a = 1.1$, and $b = 0.3$.
11) $S11$ is the Leuven time series [12].

III. Entropic Filter

This section introduces the entropic filter (EF) algorithm which is the heart of the ES system.

At the beginning, the SN will attempt to prime the predictor with basic knowledge about the temporal and statistical structure of X. This means the SN will transmit a preset number of priming samples directly to the sink (bypassing the entropic filter). After the SN transmits all the priming samples, the EF algorithm is activated for the rest of the SN's life time.

As time progresses, samples of X arrive consecutively. At each time step t, the newly added sample will alter the distribution of X and in turn its Shannon entropy $H(X)$. In other words, $H(X)$ changes every time a new sample is added to the series X. The SN will exploit the change in $H(X)$ to make the transmit/omit verdict for each sample.

Now that we have established that $H(X)$ is constantly changing with time t. We will reserve the notation $H(X)$ for the total entropy of X and define instantaneous entropy (IE) of X as follows.

Definition The instantaneous entropy (IE) of X at time t denoted by $H(X)_t$ is the entropy of X computed from the sample set $S = \{1, 2, 3, ..., t\}$.

As a new sample (s_t) arrives, the new value of the entropy is computed. It is computationally demanding to use Shannon's formula in equation 8 which requires the re-computation of X's histogram every time step. Fortunately, there is a more efficient method of computing the IE $H(X)_t$ from its own previous value $H(X)_{t-1}$ and a few other easily computed terms. This method is derived and discussed in section III-A.

Fig. 2 shows the flow chart for the EF algorithm. It consists of two parallel branches. Both branches are

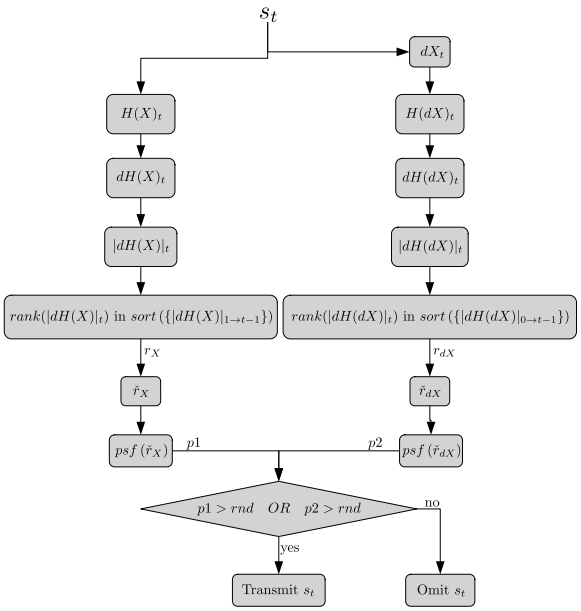

Fig. 2. The flow chart for the entropic filter.

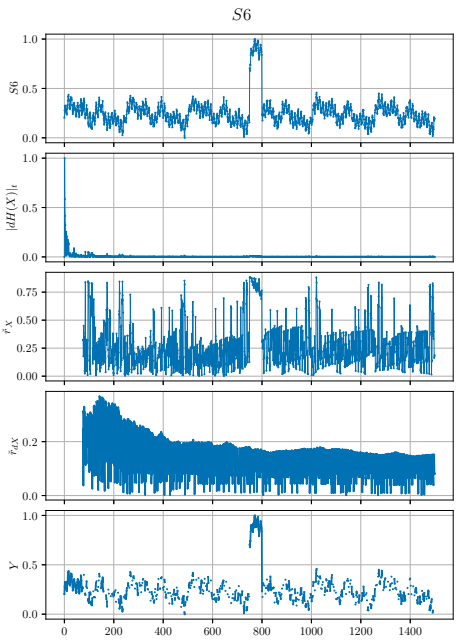

Fig. 3. From top to bottom, $S6$ as it progresses through the Entropic filtering steps.

identical except that the right branch operates on the first difference of X.

Analogous to the difference quotient in single-variable calculus, let us define the "difference of the instantaneous entropy" of X at time t (denoted by $dH(X)_t$) as the change in the IE $H(X)_t$ due to the addition of the most recent sample s_t to X.

Definition For a discrete time series X with uniform time samples represented by the vector $S = \{s_1, s_2, ..., s_t, ..., s_T\}$, the IE difference at time t is the difference between the IE of X at time t and that at time $t-1$. denoted by $dH(X)$. The IE difference is a vector with size $T-1$.

$$dH(X)_t = H(X)_t - H(X)_{t-1} \quad \forall\ t \in \{2, 3, ...T\} \tag{7}$$

After calculating the IE difference, we get its absolute value as shown in Fig. 2. This step introduces symmetry in considering samples that reduce or increases $H(X)_t$.

We then compare $|dH(X)|_t$ to its previous values and rank its value in a sorted vector of all the previously stored values. The rank of $|dH(X)_t|$ is a number r indicating the position where $|dH(X)|_t$ should be inserted if we where to keep $|dH(X)|_{1\to t-1}$ sorted. Keep in mind that $r \in \{1, 2, 3, ..., t\}$ and before passing it to the probability shaping function (PSF) (denoted by psf) it is divided by t to normalize it. The normalized rank is denoted by $\check{r}$ and the psf is discussed in more detail in Section III-B.

Fig. 3 shows $S6$ at different steps in the EF algorithm.

A. Computing Entropy

We consider the fundamental formulation for entropy which is Shannon's entropy [15] in equation 8. In the EF algorithm, the SN computes the entropy every time-step using very little computing resources. Thus, it is essential that we compute $H(X)_t$ efficiently at every time step

$$H(X) = -\sum_{x_n \in \mathcal{A}} p(x_n) \log_2 p(x_n) \tag{8}$$

Traditionally, we would start to compute $H(X)$ by empirically calculating the PMF of X (i.e., the histogram $hist(X)_t$) from all its previously collected samples. Then, we use equation 8 to calculate the entropy.

A deeper look into the traditional entropy computing method reveals considerable redundancies when calculating histograms and the entropy summation repeatedly at every time step. Consequently, we constructed an iterative computing method for $H(X)_t$ in a more efficient way by utilizing $H(X)_{t-1}$ and $hist(X)_{t-1}$.

The entropy at $t-1$ and at t is shown in equations 9 and 10.

$$H(X)_t = -\sum_{x_n \in \mathcal{A}} p(x_n)_t \log_2 p(x_n)_t \tag{9}$$

$$H(X)_{t-1} = -\sum_{x_n \in \mathcal{A}} p(x_n)_{t-1} \log_2 p(x_n)_{t-1} \tag{10}$$

Note that $p(x_n)$ is indexed by t for the same reason $hist(X)$ is also indexed by t and that is because the histogram of X changes at every time step by the addition of s_t. This means, we have to update $hist(X)_t$ as

time progresses. Fortunately, we do not have to compute $hist(X)_t$ every time from the entire series.

s_t is captured at every time step, and it will be enlisted to only one of the states in $\mathcal{A}$ denoted by x_{rt}. This means that the probability of the most recent state (i.e., at the current time) $p(x_{r_t})_t$ will increase from its previous value $p(x_{r_t})_{t-1}$. On the other hand, the probability of all the other states will decrease by a factor of $\frac{t-1}{t}$. To explain the notation of $p(x_{r_t})_t$ further, the letter r indicates the most recent state, x_{r_t} is the state of s_t where $r_t \in \{1, 2, 3, ...N\}$, and the probability of this particular state x_{r_t} at time t is denoted as $p(x_{r_t})_t$. Similarly, the state of s_{t-1} is $x_{r_{t-1}}$ and its probability at time $t-1$ and t is $p(x_{r_{t-1}})_{t-1}$ and $p(x_{r_{t-1}})_t$ respectively.

Now, let us define $\mathcal{A'}_t$ as the set of all states of X except the state in which s_t is enlisted. Thus,

$$\mathcal{A'}_t = \mathcal{A} \setminus \{x_{r_t}\} \tag{11}$$

Thus, the probabilities in $hist(X)_t$ can be updated from $hist(X)_{t-1}$ as follows.

$$p(x_n)_t = \frac{t-1}{t} p(x_n)_{t-1} \quad \text{where } x_n \in \mathcal{A'} \tag{12}$$

and,

$$p(x_{r_t})_t = \frac{t-1}{t} p(x_{r_t})_{t-1} + \frac{1}{t} \tag{13}$$

The time index t is used for two quantities; the recent state corresponding to s_t, and its probability. It is also used as a numerical value representing the total number of samples collected up to the current time t.

Before we proceed, a specific relation needs to be declared and derived for later use. If we define,

$$H(CX) = -\sum_{x_n \in \mathcal{A}} Cp(x_n) \log_2 \left(Cp(x_n)\right) \tag{14}$$

where $0 \leq C \leq 1$. Then, it follows that,

$$H(CX) = -\sum_{x_n \in \mathcal{A}} Cp(x_n) \log_2 C + Cp(x_n) \log_2 p(x_n) \tag{15}$$

$$\begin{aligned} H(CX) = &-C \log_2 C \underbrace{\sum_{x_n \in \mathcal{A}} p(x_n)}_{=1} \\ &\underbrace{-C \sum_{x_n \in \mathcal{A}} p(x_n) \log_2 p(x_n)}_{=H(X)} \end{aligned} \tag{16}$$

Therefore,

$$H(CX) = -C \log_2 C + CH(X) \tag{17}$$

By extracting the term corresponding to the most recent state x_{r_t} from the summation in equation 9 we get

$$\begin{aligned} H(X)_t = &-p(x_{r_t})_t \log_2 p(x_{r_t})_t \\ &- \sum_{x_n \in \mathcal{A'}_t} p(x_n)_t \log_2 p(x_n)_t \end{aligned} \tag{18}$$

By substituting equations 12 and 13 in 18 we get

$$\begin{aligned} &H(X)_t \\ &= \overbrace{-\left(\frac{t-1}{t} p(x_{r_t})_{t-1} + \frac{1}{t}\right) \log_2 \left(\frac{t-1}{t} p(x_{r_t})_{t-1} + \frac{1}{t}\right)}^{part1} \\ &\overbrace{- \sum_{x_n \in \mathcal{A'}_t} \left(\frac{t-1}{t} p(x_n)_{t-1}\right) \log_2 \left(\frac{t-1}{t} p(x_n)_{t-1}\right)}^{part2} \end{aligned} \tag{19}$$

For simplicity let $\frac{t-1}{t} = C$ and $p(x_{r_t})_{t-1} = R$. Then, $part1$ becomes

$$part1 = -\left(CR + \frac{1}{t}\right) \log_2 \left(CR + \frac{1}{t}\right) \tag{20}$$

and $part2$ becomes

$$part2 = -\sum_{x_n \in \mathcal{A'}_t} Cp(x_n)_{t-1} \log_2 \left(Cp(x_n)_{t-1}\right) \tag{21}$$

From equation 18 we get

$$\begin{aligned} part2 = &Cp(x_{r_t})_{t-1} \log_2 \left(Cp(x_{r_t})_{t-1}\right) \\ &\underbrace{- \sum_{x_n \in \mathcal{A}} Cp(x_n)_{t-1} \log_2 \left(Cp(x_n)_{t-1}\right)}_{=H(CX) \quad \text{from equation 14}} \end{aligned} \tag{22}$$

Then, from equation 17 we get

$$\begin{aligned} part2 = &-C \log_2 C + CH(X)_{t-1} \\ &+ Cp(x_{r_t})_{t-1} \log_2 \left(Cp(x_{r_t})_{t-1}\right) \end{aligned} \tag{23}$$

Recalling $part1$ from equation 20, $part2$ from equation 23, and recombining them in equation 19 we finally get

$$\begin{aligned} H(X)_t = &-\left(CR + \frac{1}{t}\right) \log_2 \left(CR + \frac{1}{t}\right) \\ &- C \log_2 C + CH(X)_{t-1} \\ &+ Cp(x_{r_t})_{t-1} \log_2 \left(Cp(x_{r_t})_{t-1}\right) \end{aligned} \tag{24}$$

Let $f(x) = x \log_2 x$ with following the convention $0 \log_2 0 = 0$ based on the argument of continuity as in [16]. Therefore equation 24 can be rewritten as

$$\begin{aligned} H(X)_t = &-f\left(CR + \frac{1}{t}\right) \\ &- f(C) + CH(X)_{t-1} + f\left(CR\right) \end{aligned} \tag{25}$$

As we can see in equation 25, the IE at time t can be efficiently calculated from its own value at time $t-1$. To compare on the same basis, let us rewrite Shannon's formula by using $f(x) = x \log_2 x$ as follows.

$$H(X)_t = -\sum_{x_n \in \mathcal{A}} f\left(p(x_n)_t\right) \tag{26}$$

Thus, using Shannon's formula entails N computations for $f(x)$, and $N-1$ summations while using the iterative formula in equation 25 entails only three computations for $f(x)$, five summations and four multiplications. More importantly, the iterative formula is not a function of N which means we can increase the resolution of a sensor's output without extra computing cost for the SN.

B. Probability Shaping Function

The PSF is a mapping function that allows for tuning the EF algorithm to favour samples from given ranges of $\check{r}_X$ and $\check{r}_{dX}$. This means we can design the PSF to favour low entropy samples, high entropy samples, or any other combination which can further optimise the performance of our system for a given signal.

The PSF takes the normalized rank $\check{r} = \frac{r}{t}$ of the current sample and outputs the probability of transmitting it $p = psf(\check{r})$. In this work we considered four PSFs to demonstrate their effect on the system's performance.

1) $psf1$ (high entropy): this function transmits (with probability one) high entropy samples whose $\check{r}$ is greater than or equal a predefined threshold ($th = 0.7$). Samples whose $\check{r}$ is less than the threshold will be transmitted with probability equals to $\check{r}$ itself as shown in equation 27. The intuition behind this psf can be derived from the definition of entropy itself where high entropy samples are those which introduces new structure to the series and are harder to predict by the sink thus are more worthy of being transmitted. This intuition can be also seen in [17].

$$p = \begin{cases} \check{r} & \check{r} < th \\ 1 & \check{r} \geq th \end{cases} \tag{27}$$

2) $psf2$ (low entropy): this the inverse of the high entropy psf. It is included only to establish the proper distinction in the performance of the ES system between favouring low entropy samples vs high entropy samples. The threshold in this case is 0.3. $psf2$ is defined as follows

$$p = \begin{cases} 1 - \check{r} & \check{r} > th \\ 1 & \check{r} \leq th \end{cases} \tag{28}$$

3) $psf3$ (mirrored sigmoids): this function consists of two sigmoid functions back to back. The intuition behind it is to include very low and high entropy samples in the transmission while omitting ones in the mid-low range. This psf is defined as follows,

$$p = \frac{1}{1 + e^{-(40\check{r} - 25)}} + \frac{1}{1 + e^{(80\check{r} - 6)}} \tag{29}$$

4) $psf4$ (high & random): this psf combines the high entropy function above the predefined threshold th with a uniformly distributed random function below it. It is defined as follows,

$$p = \begin{cases} random(0 \rightarrow 0.6) & \check{r} < th \\ 1 & \check{r} \geq th \end{cases} \tag{30}$$

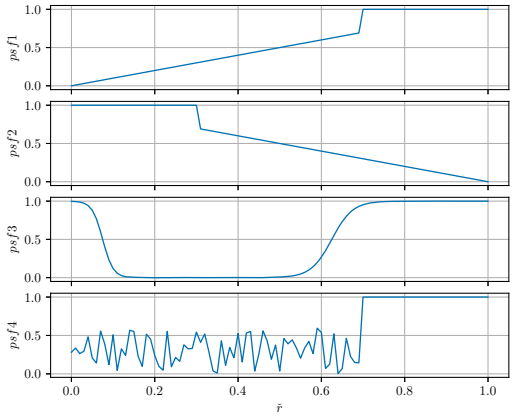

Fig. 4. The four probability shaping functions (PSFs).

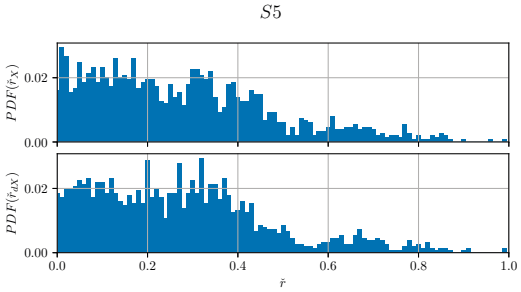

Fig. 5. Histogram of $\check{r}$ for X and dX of $S5$.

To get a better sense of what the PSF does to the population of samples, we plotted the probability density function (PDF) of the normalized rankings $\check{r}_X$ and $\check{r}_{dX}$ of all samples for $S5$ as shown in Fig. 5. We can clearly notice the skewed distribution of $\check{r}$. Most samples have low IE while few have high IE. It is important to transmit those high-IE samples and that is why three of our PSFs have a value of one for $\check{r} > 0.7$. The fourth PSF is for comparison.

Note that the histograms are constructed by comparing each sample to the past samples only. If the comparison was made among all the samples (past and future), the resulting histogram will be different and more accurate. Nevertheless, in real-time we can only access the past.

IV. Recovery

The entropic filter leaves the signal with many omitted samples. Recovering the omitted samples at the sink is the responsibility of the predictor in Fig. 1.

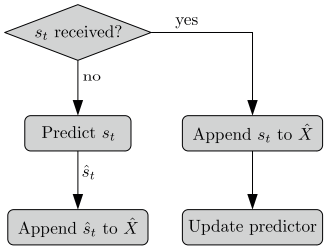

Fig. 6. Flow chart of the predictor's operation inside the sink.

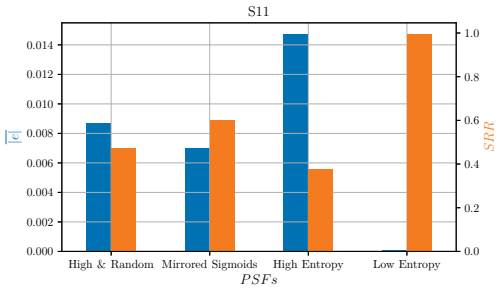

Fig. 7. $\overline{|e|}$ and SRR among all the PSFs for $S5$.

The flow chart for the predictor's operation is shown in Fig. 6. It is repeated at every time step. Thus, it is important that the choice of the predictor takes into account the available hardware resources at the sink since real-time execution is critical.

Every time step, the cycle starts with a condition. If s_t is received, it is appended directly to $\hat{X}$. Then, the predictor goes through an iteration of updating its model by utilizing the recent information in s_t plus all previous samples (received and estimated). On the contrary, if s_t is not received at its designated time, the predictor tries to recover s_t by predicting it based on its model. The predictor produces its estimate $\hat{s}_t$ and appends it to $\hat{X}$.

It is well known that the accuracy of predictions (from any predictor) decreases significantly as the number of steps ahead is increased. Fortunately, we do not have to be concerned about this here because the probabilistic nature of the EF algorithm limits the number of consecutive omitted samples to a small value (≈ 3).

V. Evaluation

The simulation was coded in python with the help of several open-source libraries such as numpy, Pandas and tensorflow.

A deep-learning predictor was chosen because it can adapt to signals on the fly. Better results may be possible if we optimize the choice of the predictor with an anticipated signal structure. For example, auto-regressive integrated moving average (ARIMA) models could be faster and more accurate for some signals [18] but, when it comes to seasonal or non-stationary ones, ARIMA models require manual tuning which is not ideal in real-time applications.

Recurrent Neural Network (RNN) in general, and Long short-term memory (LSTM) networks specifically are utilized in many applications including nonlinear system identification [19], time series prediction [20], [21] , and speech recognition [22]. Furthermore, RNN nodes form a directed graph along a temporal sequence which makes them able to capture temporal and statistical complexities of time series. Thus, we chose an LSTM network with the same hyper-parameters in all simulated signals to demonstrate its versatility.

For evaluating energy savings, we calculate the ratio between the cardinality (number of transmitted samples) of Y and that of X which is equal to T. We will call it the Samples Reduction Ratio (SRR) such that

$$SRR = \frac{\#Y}{T} \tag{31}$$

As for evaluating errors in the recovered signal $\hat{X}$ when compared to the original signal X, there are several metrics to use, however, in agreement with the arguments presented in [23] the metric we chose is the Average Euclidean Error (AEE) (sometimes called the mean absolute error, though it is not a recommended term according to [23]) denoted by $\overline{|e|}$ and calculated as

$$\overline{|e|} = \frac{1}{T}\sum_{t=1}^{T}|s_t - \hat{s}_t| \tag{32}$$

In Fig. 7, the performance of the four PSFs is compared for $S11$, we can see the difference between the "low entropy" PSF and the other three. While the "low entropy" result exhibits the lowest $\overline{|e|}$, it is not saving energy since it has the highest SRR. As for the other three PSFs, their performance is comparable with the "high entropy" function showing the lowest SRR with only a marginal increase in $\overline{|e|}$ over the "mirrored sigmoids" and the "high & random" with $\overline{|e|}$ below 0.015. Thus, with ES, we were able to recover X in real-time with 98.5% accuracy while omitting $> 60\%$ of samples from its data stream.

The results in Fig. 7 correspond only to $S11$. The results for each of the chosen bench-marking series exhibit similar outcomes.

We also look at the instantaneous absolute error $e_t = |s_t - \hat{s}_t|$ to get a sense of how errors occur over time. In Fig. 10 we can see X and $\hat{X}$ in the top plot and $\overline{|e|}$ in the bottom plot. Notice how $\overline{|e|}$ decreases over time as the prediction model gets better with incremental learning. The decreasing trend can be observed in some bench-marking signals such as $S11$ while not in others such as $S8$. The decreasing trend can be attributed to the non-increasing complexity of signals over time. The LSTM model is able to learn such complexity over time and hence, produce better predictions.

On the other hand, signals with a random component such as $S5$ to $S8$ do not exhibit the decreasing trend

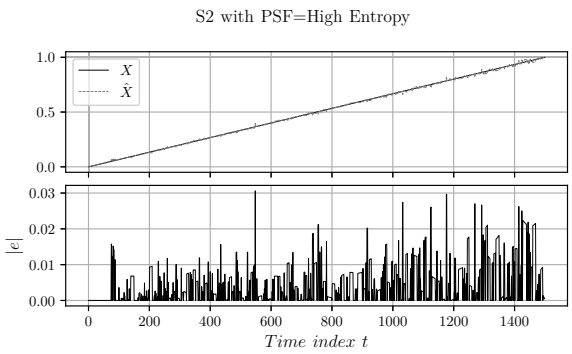

Fig. 8. Top: X and $\hat{X}$. Bottom: Instantaneous absolute error $|e|$ for $S2$.

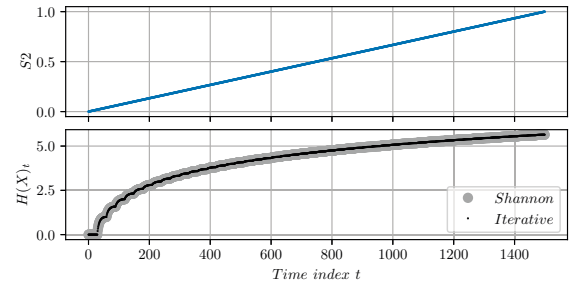

Fig. 9. Top: discrete-time series. Bottom: Shannon's entropy and its iterative form for $S2$.

in $\overline{|e|}$, because the predictor is not able to catch up with the increasing complexity of the signal over time. This is noticeable in the instantaneous entropy plots of such signals where we can see more fluctuations and no saturation as time passes. Furthermore, an increasing trend in $\overline{|e|}$ is demonstrated for $S2$ as shown in Fig. 8 for the same reason. It is because $S2$ does not cover its full dynamic range until the very end. This means new information is being added constantly to the predictor at a rate higher than its ability to learn. This can be deduced from the instantaneous entropy plot of $S2$ in Fig. 9.

Like other parameterized algorithms trade-offs are made when we tune the parameters. Varying the threshold th of $psf1$ can bias the EF algorithm towards or away from high entropy samples as desired. This has a direct effect on SRR and $\overline{|e|}$. As shown in Fig. 12, as th increases, SRR decreases because fewer and fewer samples will lie in the high entropy side of the histograms of $\check{r}_X$ and $\check{r}_{dX}$ as shown in Fig. 5. On the contrary, $\overline{|e|}$ increases with th because fewer samples mean less available information for the predictor. However, even with a very high $th = 0.95$ the mean absolute error $\overline{|e|}$ is still limited below 0.9% for $S10$ supporting our claim that most of the self-information in a time series is contained in a small subset of its samples.

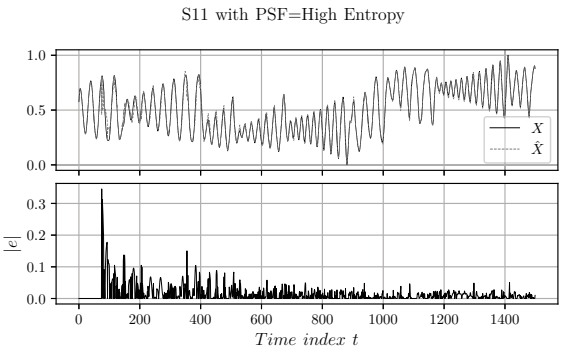

Fig. 10. Top: X and $\hat{X}$. Bottom: Instantaneous absolute error $|e|$ for $S11$.

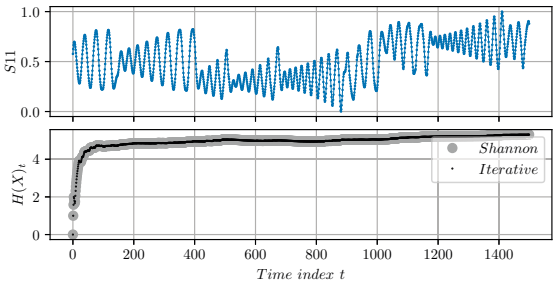

Fig. 11. Top: discrete-time series. Bottom: Shannon's entropy and its iterative form for $S2$.

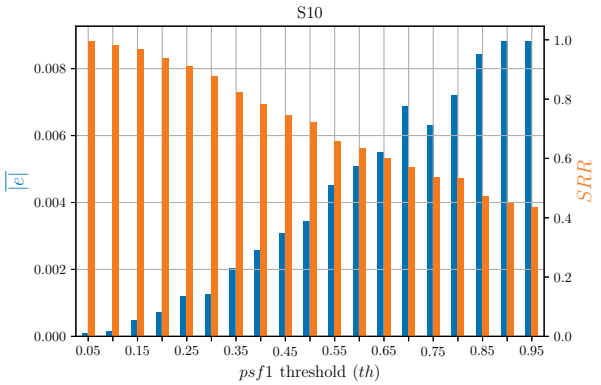

Fig. 12. Average Euclidean error ($\overline{|e|}$) and sample reduction ratio (SRR) vs the threshold (th) of $psf1$.

A. Memory Requirements

At every time step, the SN compares $\check{r}_X$ with $|dH(X)|_{1\to t-1}$ and $\check{r}_{dX}$ with $|dH(dX)|_{1\to t-1}$. This means the SN will have to store $2(T-1)$ entries by the end of its life. Also, we need to store the previous and current PMFs of X which accounts for $2N$ entries. Further memory needs are marginal and independent of the size of sensory data.

In our case $T = 1500$ and $N = 50$. Thus, most of our memory needs can be accounted for by $2(1500-1)+2\times 50 = 3098$ locations. Assuming we store all our entries in a standard floating-point format (32 b = 4 B), the required memory size amounts to ≈ 3 KB. This is highly reasonable for most currently available microcontrollers. For example, the microcontroller used in [5] (TI: MSP432P401R) has up to 64 KB of SRAM and 16 KB of flash information memory (even not including 256 KB of flash program memory).

B. Energy Savings

So far, the energy savings of the SN has been expressed in terms of the sample reduction ratio SRR. In this section, we will translate the value of the SRR to actual energy savings based on measurements from [5].

In [5], the energy consumption of the SN through one reporting cycle was measured to be ≈ 62 mJ. With the aforementioned 9-cell battery at a factor of safety 0.5, the total capacity of $2.6\times 9\times 0.5 = 11.7$ Ah $(151,632$ J). This means, the SN can report up to $\frac{151632}{0.062} = 2,445,677$ samples throughout its lifetime. However, with ES, an SRR of 0.4, which is typical in the presented results, can extend the number of samples to be reported by $(1-0.4)\times 2445677 = 1,467,406$ bringing the total number to $3,913,083$. This accounts for a 60% increase in either lifetime or reporting time resolution whichever way we decide to utilize the energy surplus.

C. Advantages

We can extend the applicability of ES with the following points. Each point represents a liberating assumption that puts ES at an advantage when compared with other approaches which have limiting assumptions in this regard.

1) There is no prior knowledge about X. Unlike what is assumed in compressed sensing schemes [24] we assume no prior knowledge about the compressing basis or the statistical structure of X. Thus, it is not possible to construct an optimal code for X. This brings ES closer to practical implementation and adds to its versatility.
2) The SN does not need a firmware update over the air during operation. While the option to update firmware during operations can improve and further tune our algorithms, most commercially available chips lack this capability. Assuming no over-the-air firmware updates during operation brings this work closer to practical implementation.
3) Communication is unidirectional from the SN to the sink. So, there is no need for a full-duplex channel. This widens the scope of applicability of this approach and adds to its spectral efficiency. Moreover, it can simplify the Medium Access (MAC) layer and reduce its required framing overhead.
4) ES can be incorporated in existing SN hardware with a firmware update along with a software update to the sink. It does not require any hardware modifications, special communications protocols, or network reconfiguration. Which means it is compliant with existing communication standards by default.

D. Limitations

SNs with multiple sensors onboard where each time series has its own EF may not have similar energy savings because the intersection between the sets of omitted samples from different sensors may not result in a significant reduction in the number of transmitted data packets. In other words, when a sample from a sensor x_{1i} is omitted, it does not necessarily mean that the sample from another sensor x_{2i} will also be omitted simultaneously. This means that the SN's transceiver will have to turn on and transmit a data packet every time step unless samples from all sensors onboard are simultaneously omitted by their respective entropic filters.

On the other hand, it is reasonable to expect different time series to exhibit similar IE trends if all signals are related to the same physical quantity (i.e., if they are correlated). Practically, if an SN is monitoring the internal temperature and pressure of a fuel tank we can expect to see no transmit-worthy samples in the temperature series if the pressure series also has no transmit-worthy samples at the same time instant. Furthermore, if there is any information-rich samples in the temperature series at a given instant, the pressure series will probably exhibit a similar behaviour simultaneously. The effect of correlations between multiple signals on the performance of their combined entropic filtering needs to be investigated in further work.

VI. Conclusion

Entropic sensing (ES) is an energy-efficient approach for wireless sensors in real-time monitoring applications. We defined the instantaneous entropy (IE) and derived an iterative formula to efficiently compute Shannon's entropy for a time series in real-time. We introduced the entropic filter (EF) which allows the wireless sensor to transmits information rich samples while omitting other samples. The EF carries marginal computing overhead, and by extension marginal energy cost on the SN. The system's demonstrated potential shows more than 60% reduction in energy consumption with no more than 2% mean absolute error in the recovered signal. The system is flexible and can be configured in many ways to fit various types of time series.

References

[1] M. Healy, T. Newe, and E. Lewis, "Wireless Sensor Node hardware: A review," in IEEE SENSORS, Oct. 2008, pp. 621–624.

[2] A. El Kouche, A. M. Rashwan, and H. Hassanein, "Energy consumption measurements and reduction of Zigbee based wireless sensor networks," in IEEE Global Communications Conference (GLOBECOM), Dec. 2013, pp. 557–562.

[3] J. Song and Y. K. Tan, "Energy consumption analysis of ZigBee-based energy harvesting wireless sensor networks," in IEEE International Conference on Communication Systems (ICCS), Nov. 2012, pp. 468–472.

[4] J. Lu, H. Okada, T. Itoh, T. Harada, and R. Maeda, "Toward the World Smallest Wireless Sensor Nodes With Ultralow Power Consumption," IEEE Sensors Journal, vol. 14, no. 6, pp. 2035–2041, Jun. 2014.

[5] M. A. Ibrahim, G. Hassan, H. S. Hassanein, and K. Obaia, "Wireless Sensing for Ground Engaging Tools," in IEEE Global Communications Conference, Taipei, Taiwan, Dec. 2020.

[6] F. Marcelloni and M. Vecchio, "A Simple Algorithm for Data Compression in Wireless Sensor Networks," IEEE Communications Letters, vol. 12, no. 6, pp. 411–413, Jun. 2008.

[7] A. Y. Carmi, L. Mihaylova, and S. J. Godsill, Eds., Compressed Sensing & Sparse Filtering, ser. Signals and Communication Technology. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[8] D. L. Donoho, "Compressed sensing," IEEE Transactions on Information Theory, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[9] M. S. Asif and J. Romberg, "Dynamic updating for sparse time varying signals," in 43rd Annual Conference on Information Sciences and Systems, Mar. 2009, pp. 3–8.

[10] D. Angelosante, G. B. Giannakis, and E. Grossi, "Compressed sensing of time-varying signals," in 16th International Conference on Digital Signal Processing, Jul. 2009, pp. 1–8.

[11] A. Korlekar, U. Pacharaney, and R. K. Gupta, "Energy efficient information gathering in wireless sensor networks using compressive sensing," in International Conference on Information Communication and Embedded Systems (ICICES), Feb. 2016, pp. 1–8.

[12] "McNames Data Sets," https://web.cecs.pdx.edu/~mcnames/DataSets/index.html.

[13] S. R. Abbas and M. Arif, "New Time Series Predictability Metrics for Nearest Neighbor Based Forecasting," in IEEE International Multitopic Conference, Dec. 2006, pp. 100–105.

[14] M. Hénon, "A two-dimensional mapping with a strange attractor," Communications in Mathematical Physics, vol. 50, no. 1, pp. 69–77, Feb. 1976.

[15] C. E. Shannon, "A mathematical theory of communication," The Bell System Technical Journal, vol. 27, no. 3, pp. 379–423, Jul. 1948.

[16] T. M. Cover and J. A. Thomas, Elements of Information Theory 2nd Ed., 2nd ed. Wiley Interscience, 2006.

[17] M. C. Shewry and H. P. Wynn, "Maximum entropy sampling," Journal of Applied Statistics, vol. 14, no. 2, pp. 165–170, Jan. 1987.

[18] K. Miranda, V. M. Ramos R., and T. Razafindralambo, "Using efficiently autoregressive estimation in wireless sensor networks," in International Conference on Computer, Information and Telecommunication Systems (CITS), May 2013, pp. 1–5.

[19] D. Liu, "Open-loop training of recurrent neural networks for nonlinear dynamical system identification," in IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222), vol. 2, Jul. 2001, pp. 1215–1220 vol.2.

[20] S. D. Kumar and D. Subha, "Prediction of Depression from EEG Signal Using Long Short Term Memory(LSTM)," in 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Apr. 2019, pp. 1248–1253.

[21] M. A. I. Sunny, M. M. S. Maswood, and A. G. Alharbi, "Deep Learning-Based Stock Price Prediction Using LSTM and Bi-Directional LSTM Model," in 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES), Oct. 2020, pp. 87–92.

[22] J. Jo, S. Hwang, S. Lee, and Y. Lee, "Multi-Mode LSTM Network for Energy-Efficient Speech Recognition," in International SoC Design Conference (ISOCC), Nov. 2018, pp. 133–134.

[23] X. R. Li and Z. Zhao, "Measures of performance for evaluation of estimators and filters," in International Symposium on Optical Science and Technology, O. E. Drummond, Ed., San Diego, CA, USA, Nov. 2001, p. 530.

[24] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak, "Compressive Wireless Sensing," in Proceedings of the 5th International Conference on Information Processing in Sensor Networks, ser. IPSN '06. New York, NY, USA: ACM, 2006, pp. 134–142.