Compressive Sensing Optimization Over ZigBee Networks

Carlo Caione*, Davide Brunelli† and Luca Benini*
*University of Bologna, Viale Risorgimento 2, Bologna, Italy
{carlo.caione|luca.benini}@unibo.it
†University of Trento, Via Sommarive 14, Povo (TN), Italy
davide.brunelli@disi.unitn.it

***Abstract*—Efficient data aggregation and compression in sensor networks is becoming fundamental with the increase of the number of nodes in the network. Although several data aggregation and compression techniques have been proposed in the literature only few of them can perform in-network compression and can extend lifetime without prior knowledge of the sensed data or without a central coordination. In this paper we consider a scenario where a wireless sensor network (WSN) exploits ZigBee protocols for smart building application. We study a classical gathering scheme and a distributed compressive sampling approach. We discuss limitations and we propose a new distributed mixed algorithm for in-network compression. With this algorithm each node takes a decision about which scheme to adopt aiming at the reducing the number of packets to transmit. We are interested in scalability of this new method and lifetime of the system with respect to the increase of network dimension. Simulations are performed using real data sets and results show that the use of this algorithm permits to obtain longer network lifetime with small computational complexity. The performances of the algorithm are also investigated when some sensor parameters are modified and sporadic readings rise in the network.**

I. Introduction

This paper addresses the issue of data gathering in a large-scale wireless sensor network. Sensor networking is an emerging technology that promises the ability to monitor and manage buildings using wireless sensor node spread into environment. These nodes are usually small and inexpensive devices with severe energy constrains. According to the field of use, the number of these deployed sensor nodes could be on the order of hundreds or thousands. This huge number of devices, communicating sensor data to a central sink through multi-hop routing from individual sensors, arises the problem to gather data in an efficient way[14].
Successful deployment of such kind of networks is strictly connected to the lifetime of the network that becomes a critical parameter[13] because of the high energy consumption of data transmission. In this scenario data gathering techniques are developed to ensure data collection at a minimum energy loss. The problem to transmit and store information from one point to another inside a large scale distributed network of data sources is well known, but the real challenge is to efficiently perform aggregation and gathering inside the network. The huge number of nodes introduces, in fact, several problems to manage the big amount of data within the deployed network. Moreover the computational power needed to process and to store networked data is too demanding for a single node in case a single aggregation point is used inside the network.
For these reasons the global communication and energy cost reduction is obtained through in-network decentralized compression to increase the energy efficiency of data gathering techniques preserving the accuracy of data in the reconstruction phase[15]. One of the most explored strategies in this field is to take advantage of the correlation among sensor data to build distributed compression techniques, such as entropy coding, to reduce traffic[16]. Unfortunately this approach is not useful in WSN applications where a priori knowledge of the joint distribution quantifying the sensor correlation is not available. Moreover these distributed coding techniques are not suitable for small resource constrained sensors with limited computational power.
Compressive sampling (CS) is a novel data compression technique that can overcome some of the limitations of the classical compression schemes[12]. CS is universal in the sense that it does neither require any priori information about data correlation nor a central controller for encoding, ensuring decentralized in-network compression. The novelty of CS is brought by the use of a quasi-random compression matrix used to compress original data X into a shorter version Y whose length depends on the sparseness of the signal X. The compression can be performed jointly with routing and transmission, while decoding affects only the sink.
However, while CS is a very powerful tool because it shifts compression and computations from single nodes to the entire network (and sink), there are up to now significant limitations to the large scale adoption of CS in WSN because of increased power consumption and packets transmission when real wireless sensor networks are taken into account.
In this paper our goal is to investigate the performance of both CS and a classical data gathering techniques with increasing number of network nodes. We consider a real case ZigBee WSN for temperature monitoring. Our main contribution is the design of a new adaptive mixed algorithm for data transmission, designed to minimize the power consumption and packets transmission according to network status. In addition we study how the relay of incompressible additional data can affect the performance of this technique.
The paper is structured as follows. In Section II we present

related works on CS applied to WSN and in Section III a brief description of the mathematical background on CS is given. In Section IV scenario and network model are described while in Section V data aggregation and compression schemes used in the simulation are presented. In Section VI our new algorithm is introduced and in Section VII results are presented. In Section VIII the conclusions.

II. Related Work

The problem to perform data gathering and compression inside a WSN is an interesting and challenging problem, widely investigated in the literature. A very well-known technique for in-network compression is distributed source coding, based on Slepian-Wolf coding[10] theory. Since sensors of the network presumably observe the same phenomenon, it is very likely that acquired data have high inter-signal correlation. This correlation property is used by distributed source coding techniques to save on communication costs, allowing each node to encode each own signal as it was jointly encoded and then reducing the number of bits needed for signal representation. DISCUS[1] is a constructive algorithmic framework for collaborative networked signal processing able to achieve compression without inter-node communication. However this framework, and distributed source coding in general, requires that decoder is aware of global correlation structure behind the network, that is a difficult task to achieve in large-scale WSN. An other slightly different approach is in [2] where the joint entropy is empirically obtained as function of the distance between nodes. In this case entropy is taken as measure to compare distributed source coding with two other schemes: routing driven compression in which data is sent along the shortest path toward sink, and compression driven routing where routing is performed to achieve maximum aggregation. Even though a mixed version of this two methods has been proved by author to be more energy-aware than classical distributed source coding, it requires a special routing scheme inside the network and in addition the algorithm needs a global coordination among nodes to promote cluster-heads.

As previously stated, CS is becoming a valid tool and framework for data compression in large-scale network. In [3] authors establish a parallel between CS and the Slepian-Wolf theorem creating a new theory for distributed compressed sensing able to exploit both intra- and inter-signal correlation. This approach is different from ours because they mainly focus on measurements rate needed to encode signal creating a new joint sparsity model. A very interesting application of CS is presented in [6] where sparse random projections are used to drastically reduce the communication cost. Authors use a sparse random matrix to compress data and they investigate conditions needed to recover an approximation of the original signal with minimum error. The aim is to obtain a signal reconstruction querying an arbitrary number of sensors, so they do not explore typical network configuration with a central gateway acting as a data collector. This common network configuration is examined in [4] and [5] where the sparse random matrix is built during the routing toward the sink. While in the former work a centralized routing algorithm has been developed to better choose nodes in the path toward the collector, in the latter real and synthetic signal are reconstructed at the sink, evaluating the quality of the reconstruction. Results show how the use of sparse random matrix with real signals does not outperform classical compression schemes. Moreover both works do not take into account, real technological factors and there is no indication on scalability of the proposed techniques on networks of different size.

A brief comparison between CS and classical gathering scheme is proposed in [7]. Authors adopt a gathering scheme based on CS to perform delay and capacity analysis in a large network using NS-2 simulator. A new technique for identification and compression of sporadic readings is presented, together with a mathematical formalism able to incorporate signal recovery with these abnormal readings. Their approach to compression is similar to ours, but they do not investigate new compression strategies mixing classical gathering scheme and CS. Moreover the focus of simulations is on output-input interval without stressing energy consumption and packet traffic inside the network.

In our work we address the compression problem in a large-scale WSN first analyzing performance of a classical gathering scheme and CS in a real network subject to limitations brought by communication protocol. Then we propose a mixed algorithm able to reduce the overall packet transmission ensuring data compression. This is a fully distributed algorithm in which each node has enough information to take a proper decision about data compression, extending the lifetime of the network. We only require that each node is aware of some parameters to decide between CS or a simpler gathering scheme. The goal of the work is to extend the lifetime of the overall network and to reconstruct the original signal with good quality, limiting the number of packets circulating inside the network. To the best of our knowledge this is the first paper addressing such a mixed algorithm. We do not focus on reconstruction loss of efficient compressed sampling approximations, but our interest is on lifetime of the network when CS is used to compress data toward a central sink. For this reason the analysis is not performed on a network with fixed dimension, but we investigate the scalability of this approach in networks of increasing size. Finally, we also analyze the performance of the algorithm when sporadic incompressible data is sent jointly with compressed payload.

III. CS: Mathematical Background

Compressive sensing (CS) is a technique used to compress data to obtain a representation for signal that is smaller (in terms of samples) than the original one. The mathematical theory behind CS guarantees that, under certain conditions, we can recover the original signal with high probability through the resolution of an optimization problem[11]. These conditions claim that if a signal is sparse in one basis, it can be recovered from a small number of projections onto a second basis that is incoherent with the first.

Specifically a generic signal $x \in \mathbb{R}^N$ indexed as $x(n), n \in$

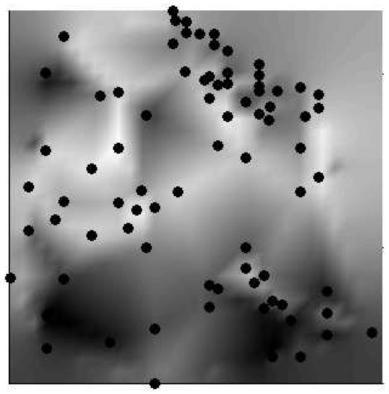
Fig. 1: Temperature field from EPFL SensorScope. Black dots are station locations in the real deployment.

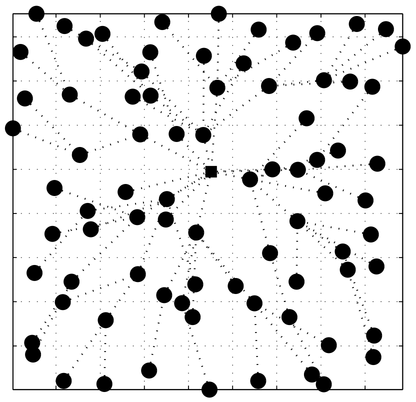
Fig. 2: Example of the considered WSN with 9×9 sensors.

$\{1, 2, \ldots, N\}$ can easily be seen as the networked data vector created by the sensor readings of a network with N nodes. In case of 2D signals a generalization can be easily obtained, but for the sake of simplicity from now on we address only 1D signals.
We assume that this signal x can be sparsely represented in a certain basis $\Psi = [\psi_1, \psi_2, \ldots, \psi_N]$; that is:

$$x = \Psi a \tag{1}$$

where a is a K-sparse representation of x such that $\|a\|_0 = K$ column vectors $N \times 1$ chosen from Ψ are enough to represent the original signal x. Matrix Ψ is called sparse basis matrix and it can be constructed from various bases: wavelets, DCT, Gabor bases, curvelets, etc...
The compression of x is performed through a further measurement matrix $\Phi = [\phi_1, \phi_2, \ldots, \phi_N]$ of size $M \times N$, with $M < N$. If y is the compressed version of x we can write:

$$y = \Phi x \tag{2}$$

where y is an $M \times 1$ column vector. That is we encode and measure M projections y of the signal x onto a second set of basis functions $\{\phi_n\}$.
If these two matrix Φ and Ψ are incoherent (elements of the matrix Ψ are not sparsely represented in the basis $\{\phi_n\}$) and M is large enough, CS theory assures that we can recover a (and then indirectly x) from measurements y. Interestingly, independent and identically distributed (i.i.d.) Gaussian or Bernoulli/Rademacher vectors provide a universal basis that is incoherent with any given Ψ with high probability[12].
Then putting together (1) and (2) we have:

$$y = \Phi\Psi a = \Theta a \tag{3}$$

where Θ is known as the holographic basis.
The problem to recover a from y as in (3) is ill-posed as the number of equations M is smaller than the number of variables N. Thus the recovery can be only achieved using optimization, searching for the resolution of:

$$\hat{a} = \text{arg min}\|a\|_1 \;\text{ s.t. }\; y = \Theta a \tag{4}$$

This problem, known as basis pursuit, can be successfully solved with classical linear programming techniques only if we have $M \geq cK$ measurements, where c is an oversampling factor. Obviously the reconstruction error depends on the number of measurements and in general it decreases for increasing values of M (for $M = N$ the problem is solvable through matrix inversion). Data can be reconstructed with high probability from $M = 3K \sim 4K$ random measurements.

In CS, compression and decompression are two totally different and separated phases: the former in a WSN is accomplished in the network through aggregation during transmission toward the sink, while the latter is only performed by sink through resolution of the problem in (4). It is worth noting that the sparse matrix Ψ is used only at the decoder and it is not needed for compression in which just the knowledge of Φ is enough.

IV. Scenario and Network Model

In this Section we describe the scenario and the motivations behind our choices. Advances in sensor technology and the miniaturization trend are pushing development of new technologies in WSN that are becoming a very useful tool for environmental monitoring, smart building management and in general for acquisition of data from a large set of sensor spread into the environment. Moreover in resource-constrained node the more power hungry activity is data transmission, so many low-power transmission protocols have been developed to increase lifetime of the network. For all these reasons in this paper we deal with a real case WSN deployment for monitoring over a large area through ZigBee sensor nodes. ZigBee appears to be up to now one of the best communication protocols among power-aware protocols suite that can guarantee energy saving on such a large-scale WSN. Moreover performances of CS are strictly related to the characteristics of the original signal and in particular to its

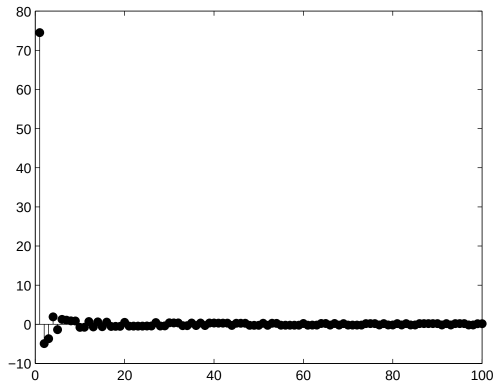

Fig. 3: Top 100 DCT largest coefficients for temperature field.

sparsity. Furthermore in literature it is well known as large differences exists between synthetic and real signals[4]. So in this paper we work with temperature monitoring that is likely to be of interest and suitable for large-scale WSN.
In our tests the sets of measurements are taken from LUCE[8] deployment of EPFL SensorScope WSN, reporting ambient temperature recorded by 100 weather station deployed on the EPFL campus. The temperature field in Fig. 1 is reconstructed using GPS data and rescaled on square region through extrapolation; data is collected at midnight on December 22, 2006. We consider N nodes to be deployed in a square area. If the sensors are geographically placed in a uniform fashion the sensor locations can be viewed as sampling locations greatly simplifying calculations in CS [9]. So the area of interest is split into N square cells and we suppose that in each square is placed a sensor node in a random position. The value read from each sensors is expected to be very similar to that one present nearby, assimilating the temperature recorded in just one point to that one of every point in the square.
In the network each node can communicate with just one other node within communication range R_c. In our simulation R_c is set in such a way to permit communication among every nodes. Moreover a geographic routing [4] is adopted to forward data toward the sink which is conventionally placed at the centre of the network. This geographic routing approach forces the node to forward data to the node within R_c that is closer to the sink than current node. The sink is supposed to be connected for monitoring to a remote station through GSM connection, this is why in-network compression is a need: it is not opportune to send uncompressed data because of cost and throughput of connection. The in-network compression is also adopted to drastically reduce the amount of packets circulating in the network, extending the lifetime of the system.
In Fig. 2 we show a network example built with the rules above on geographical routing. Furthermore temperature is not a signal subject to fast changes, but dynamic is quite slow in time. Thus to save energy, temperature is monitored by nodes once per minute, spending the rest of time in sleep mode preserving battery power.
Lifetime and energy saving are not the only goals we are interested in, we want to be also sure about signal reconstruction at decoder. As we said in Section III the reconstruction quality is function of the number of measurements. In particular, for a constant K, the error depends on the oversampling factor c. CS theory states that $c(S) = O(\log(1/S))$ where S is the sparsity rate such that $K = SN$ if the sparsity scales linearly with N. Fortunately in literature[3] we can find more practical characterizations for c such as $c(S) \approx \log_2(1 + S^{-1})$. Then considering that for random measurements has to be $M \geq cK$ we have:

$$M(N) \geq \log_2(1 + S^{-1}) \cdot S \cdot N \tag{5}$$

Equation (5) specifies that the number of random measurements needed to recover the signal is function of the length of the signal N and sparsity rate S (considered as a constant). With regard to N this means that the bigger our network, the larger the number of measurements we need. On the contrary S is an inner property of the signal and it defines how many nonzero coefficients we have in the transform coding defined by sparse basis matrix Ψ with respect to N. Obviously S strictly depends on the signal to encode, and it is different according to the signal field of interest.
Since our aim is to monitor ambient temperature, so S has to be obtained by such a kind of spatial monitoring. To obtain the value of sparsity rate S we can use the temperature field, apply to it a DCT transform (or any other transform coding) and then evaluate the number of coefficients significantly different from zero. The choice of DCT for our purposes comes from the consideration that the spatial signal in Fig. 1 appears to be smooth and then suitable for DCT analysis.
Fig. 3 shows the coefficients of DCT when original data is rescaled on a 32×32 grid. There are only few coefficients whose absolute value is significantly larger than 0.6 accounting for approximately 5.0% of the total coefficients. Although the rest of coefficients are not strictly zero we can confidently set $S = 0.05$ and take this value as a constant for this kind of signals during our simulations.

V. Data Gathering and Aggregation

In Section IV we have described the scenario and the network model, pointing out some characteristics of the signal in terms of sparsity. In this Section we are going to define the data gathering mechanisms implemented in our network. when a real communication protocol (ZigBee) is used for radio communication. Then in the next Section we introduce a new algorithm that accounts of these problems come up with a much better compressive gathering solution.

A. *Pack & Forward (PF)*

This is a gathering and aggregation scheme that is a slightly modified version of the classical relay mechanism generally present in WSN. In the typical relay scheme, each node that receives a packet being in the route path toward the gateway forwards it to its parent; this generates a huge waste of communication power proportional to the number of packets to relay. To limit the power consumption is mainly needed to

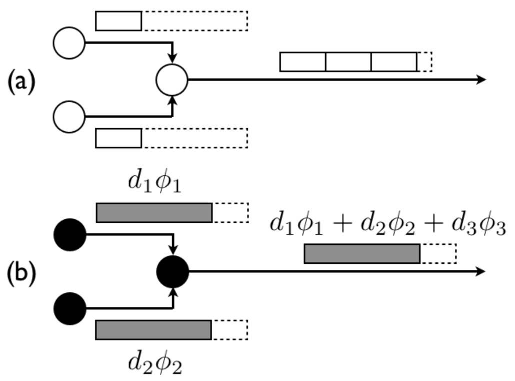

Fig. 4: Dashed rectangle is the ZigBee packet. (a) PF: data incoming in each node are packed through aggregation. (b) DCS: the outgoing data size is independent from the number of children nodes.

reduce the number of packets sent out from each node. It is well known in fact as the more power-hungry operation in a WSN node is the transmission of data over the air.

Pack & Forward (PF) is a more energy-safe strategy in which each node tries to encapsulate data in the smallest possible number of packets.

As seen in Section IV, in the network under test we adopt a geographic routing. Considering that nodes are fixed in space, we can suppose that each node is aware of the number of children nodes it has. Therefore instead of sending every message to the data sink, intermediate nodes delay messages until they have received (all or some) messages from their children nodes, compute an aggregated value of all these values, and then forward only a single message with the aggregated value. A representation of this mechanism is represented in Fig. 4a. This aggregation scheme allows to save on the total number of network transmissions.

The decoder needs two information to reconstruct the signal when such a kind of aggregation is performed: (i) the sensor reading (supposed to be a scalar) and (ii) the identification code (ID) of the sensor node. For the sake of simplicity we assume that the sink knows the placement of nodes in the network and it is able to match an ID with a spatial location. Common sizes for these data fields are: at least 1 byte for scalar data and 8 bytes for ID. The scalar reading is thought to be at least an 8 bit value, while EUI64 (64 bits) is a reasonable choice as ID since it is a unique number easily accessible by node itself and hardcoded in network stack, but we have also performed simulations with different values. ZigBee packet according to the specifications has a 127 bytes payload. The ratio between the number of bytes used for information about data and location, and the maximum payload allowed by transfer protocol, defines the saving obtained by PF strategy. According to this method the number of packets sent by each node, without fragmentation, is equal to:

$$P_k^{PF} = \lceil((N_{tch}+1)\cdot(B_{ID}+B_{data}))/B_{payload}\rceil \qquad (6)$$

where P_k^{PF} is the number of the outgoing packets from node k, N_{tch} is the number of nodes in the path tree doing relay on node k, B_{ID} and B_{data} are respectively the required bytes for ID and data representation and $B_{payload}$ is the number of bytes admittable by ZigBee payload.

It is clear how the parameter N_{tch} is strictly related to the network size and it depends also on the distance of the node from the central sink. Therefore when the number of nodes inside the network becomes bigger, the number of total transmission obviously grows up.

B. Distributed Compressed Sampling (DCS)

In Section III we have introduced the mathematical bases for CS. These concepts can be easily adapted to perform a powerful form of in-network compression, originating the so called Distributed Compressed Sampling (DCS). As seen in the introduction, DCS (and CS) is an innovative approach to encoding and compression because it does not require any specific prior signal knowledge. Moreover the method is fully decentralized since compression is performed during the relay of data toward the sink.

The compression phase consists of a projection of the signal x onto a set of basis function $\{\phi_n\}$. From a practical point of view the compression phase is the multiplication of a scalar for a random vector. Equation (2) in Section III can be seen in fact as:

$$y_i = \sum_{j=1}^{N} \phi_{ij} x_j \qquad (7)$$

where y_i with $i = 1, \dots, M$ is the i-th element of the compressed output vector, ϕ_{ij} is a random element of the measurements matrix and x_j is the scalar data sensed from the j-th sensor node. In this form DCS is well suited to be used in a decentralized fashion since each node simply adds its own contribution to the global sum. Algorithmically the compression is performed in two different steps:

1) Each of the N sensors locally computes the M elements of the random vector $\phi_j = \{\phi_{ij}\}_{i=1}^{M}$ using its own address (or any other unique number known also by decoder) as seed for the pseudo-random sequence.
2) The j-th sensor multiplies its sensor reading for the vector ϕ_j just computed obtaining a new $M \times 1$ vector. Then the sensor waits until it receives by each of its children nodes the compressed vector. Once received, it simply sums all the vectors together and sends the newly computed vector to its own parent.

A visual explanation of the process is in Fig. 4b. To decompress data the decoder needs only the resulting y vector and every ϕ_j vector to be able to reconstruct the whole Φ matrix used in the optimization problem, as seen in eq. (4).

Differently from PF, the number of packets sent is constant for the entire network and depending only on the dimension M of the random vector used for compression. Therefore in this case the number of packets sent by a node k using DCS is:

$$P_k^{CS} = \lceil(M \cdot B_{data})/B_{payload}\rceil \qquad (8)$$

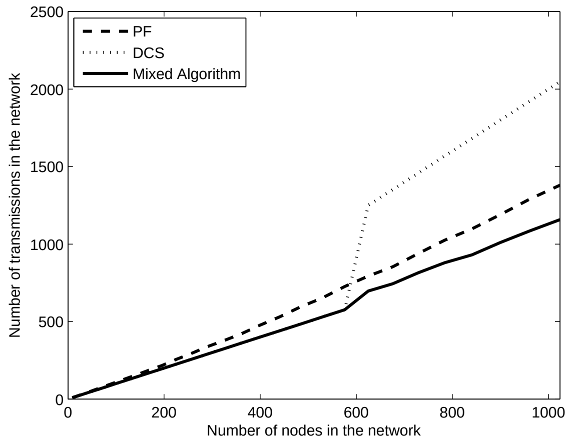

Fig. 5: Number of transmissions in the network vs network size: comparison among PF, DCS and Mixed Algorithm.

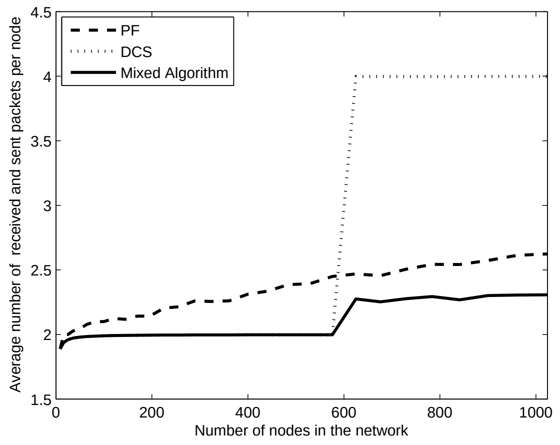

Fig. 6: Average number of packets per node vs network size: comparison among PF, DCS and Mixed Algorithm.

where each symbol has the usual meaning.
It is important to point out that M, as seen in expression (5) is a function of the number of nodes N. Differently from the case in subsection V-A the number of outgoing packets is the same for every node in the network and it is function of the network size. The limitation with this approach is that when the value of P_k^{CS} increases, it increases for all the nodes in the network, causing a huge amount of additional packets in the network and then more power consumption, narrowing lifetime.

VI. Mixed Alghoritm

The idea behind the aggregation scheme proposed in this Section is to achieve a tradeoff between the two mechanisms previously described, trying to overcome their limitations with a new distributed alghoritm.
As seen in subsection V-A, and specifically in equation (6), the performance of PF scheme, in terms of P_k^{PF}, is strictly related to parameter N_{tch}, that is bigger for nodes proximal to the sink. On the contrary, DCS performance is only connected with parameter M that is proportional to network size. Therefore it is not possible to identify a mechanism that could be considered better with respect to the other one for all the situations.
The basis of this mixed algorithm is that each node can independently take a decision about what kind of gathering scheme between the two illustrated to adopt aiming to reduce the number of transmitted packets. Algorithmically, this decision is taken in three different steps:

1) If the node receives a packet from one of its children nodes which has been compressed with CS, it is forced to adopt DCS scheme, and then use CS to compress data, performing aggregation as described in subsection V-B.
2) If the node receives all the packets from its children nodes and none is compressed using CS, it chooses to adopt PF or DCS aiming to send the lowest number of packets.
3) If it is not possible to take a decision, CS is always performed.

Step 3 is useful in those situations in which the number of outgoing packets is the same for both the schemes: in this case the choice of CS is taken to have compressed data to the central gateway.The choice done in step 2, can be accomplished locally by each node.
Considering in fact the equation (6) we can rewrite this equation as:

$$P_k^{PF} = \left\lceil \left(\sum_{j=1}^{H} B_j + (B_{ID} + B_{data}) \right) / B_{payload} \right\rceil \tag{9}$$

where H is the number of children nodes and the new term B_j refers to the number of bytes of payload received by the j-th child node. In this way the node has all the informations it needs to take the proper decision: M, $B_{payload}$, B_{ID} and B_{data} are parameters we can assume known at the bootstrap, while H and B_j are obtained during operation.
Considering equations (9) and (8) it is straightforward for a node to make a decision aiming to the reduction of the outgoing packets. In formulas we can write:

$$P_k = \begin{cases} P_k^{CS}, & \text{if } P_k^{CS} \le P_k^{PF} \\ P_k^{PF}, & \text{if } P_k^{CS} > P_k^{PF} \end{cases} \tag{10}$$

where P_k is the number of packets actually sent by node.
Even if the mixed algorithm is distributed in the sense that it does not need a global coordinator and every decision is made locally, the global behavior is influenced also by other parameters such as B_{ID} and B_{data} that are chosen during network and sensor design. During the simulation we investigate these parameters and in which way they can influence our algorithm.

A. Sporadic Readings

One of the main purposes of the WSN is to monitor sporadic readings. These events are not periodic or programmed like

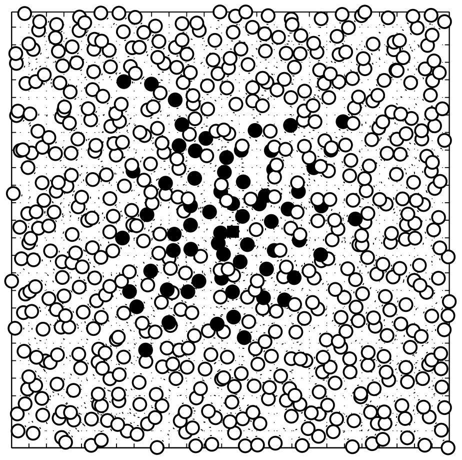

Fig. 7: Black circles are nodes performing DCS. The white ones PF.

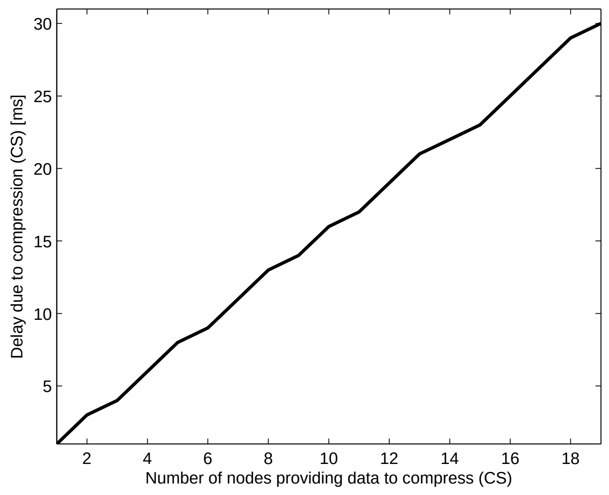

Fig. 8: Delay created by compression routing on Ember EM250.

temperature monitoring, but determined by an instantaneous need concerning one or a small group of specific nodes.

A real case related to our scenario could be the need to capture, in case of emergency, an image taken by a camera on the node. This data has to be moved from the node to central sink together with temperature data. We can think to this amount of data as incompressible or already locally compressed by the node.

This additional information has to be sent inside the packet payload, increasing its dimension for those nodes lying in the routing path of the original node.

In this case the equations (8) and (9) are slightly modified:

$$P_k^{CS} = \lceil ((M \cdot B_{data}) + B_{incomp}) / B_{payload} \rceil \tag{11}$$

and

$$P_k^{PF} = \left\lceil \left(\sum_{j=1}^{H} B_j + (B_{tot}) \right) / B_{payload} \right\rceil \tag{12}$$

where $B_{tot} = B_{ID} + B_{data} + B_{incomp}$ and B_{incomp} is the amount in bytes of the incompressible data.

The mixed algorithm is still perfectly valid as described in the previous Section paying attention to consider also the contribution of B_{incomp}. In the following Section we address also the problem to study the behavior of our algorithm in presence of this extra-payload.

VII. Simulation Results

All the simulations are performed using a custom framework written in MATLAB. Each node is assumed to have a temperature sensor and a micro-camera onboard.

To evaluate the performance of the mixed algorithm, parameters are set to standard values: $B_{ID} = 8$ bytes and $B_{data} = 1$ byte.

In Fig. 5 the total number of transmissions in the network is reported when the size of the network varies from 9 to 1024 nodes, while in Fig. 6 it is possible to see the average number of received and sent packets per node.

For PF the number of transmission slowly increases with N, while DCS shows a huge increase in traffic for a specific network dimension when the value of P_k^{CS} goes from 1 to 2 causing an explosion in the number of transmissions and doubling the average number of transmitted and received packets per node. This explosion is due to the doubling in the number of sent packets for DCS affecting each of the network nodes; PF does not show this behavior since the number of packets is not influenced by M factor, as seen in equation (6). This is obviously undesired, since more packets circulate, more power consumption we have in the network, narrowing lifetime. At the opposite, simulations show how the adoption of the mixed algorithm brings a practical advantage, reducing the traffic inside the network. This is clearly visible since the line referring to the algorithm is always below the other two. For small networks since the performance of PF and DCS are the same (same number of outgoing packet for each node), the algorithm allows the compression for all the nodes, justifying the overlap between mixed algorithm and DCS. For a certain value of N, called critical and indicated with N_{crit}, the value of P_k^{CS} (equation (8)) increases, making the compression with CS not convenient for nodes distant from sink.

It is interesting to see what happens inside the network when the mixed algorithm is used and $N^* > N_{crit}$. Fig. 7 is an illustration of the network for this fixed N^*. Since P_k^{PF} is strictly function of N_{tch}, nodes proximal to the sink (or better with high N_{tch}) compress data with DCS, creating an inner zone round the sink where DCS takes place while the more distant nodes can perform simple aggregation with PF.

The nodes at the frontier in this region are those ones delegate to perform compression of data coming from the leaves of the network. They are the nodes that perform the most intensive computational work inside the network.

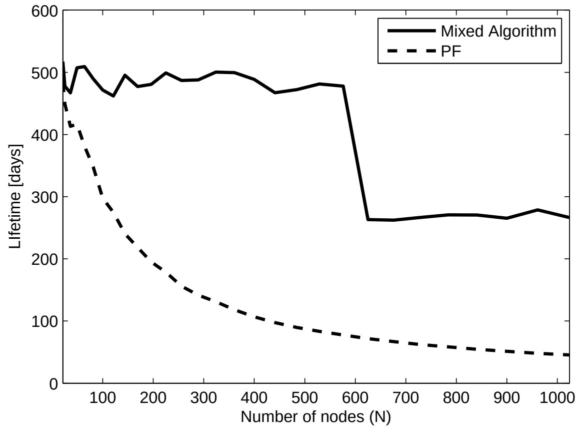

Fig. 9: Lifetime of the network: comparison between PF and Mixed Algorithm ($B_{ID} = 1$ byte, $B_{data} = 8$ bytes).

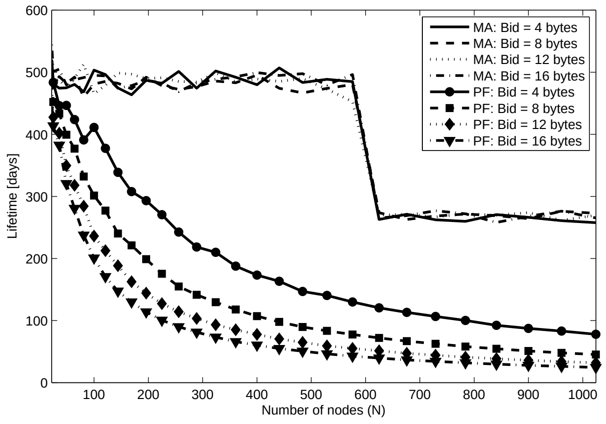

Fig. 10: Lifetime of the network: comparison between PF and Mixed Algorithm (MA) varying B_{ID}.

To verify that small nodes can perform compression without creating too high delay in communication, we have tested the compression routine on real small nodes and measured delay. The hardware used is an Ember EM250, a ZigBee System-on-Chip that combines a 2.4GHz IEEE 802.15.4 compliant radio transceiver with a 12MHz 16-bit microprocessor equipped with 128Kb of flash memory and 5Kb of RAM.

The compression routine tested is the C implementation of the equation (7) using $M = 100$ and N varying from 1 to 20. The results shown in Fig. 8 are encouraging: with simple hardware, the delay increases linearly but it is always small enough not to affect too much network performance.

Using the same hardware we have also measured the power consumption involved in compression. This has been done using a precision resistor to evaluate the current absorbed by EM250. The current absorbed, according to the experiment, is around $9.3mA$ during compression. This value together with the electrical characteristics of EM250[17] has been useful to define the lifetime of the system. Furthermore we assume that each node is powered by a $500mAh$ battery. In Fig. 9 is presented the lifetime, averaged over 20 measurements, of the network comparing our algorithm with the classical gathering scheme. The lifetime is calculated as the time until the death of the first node. As seen in the plot, mixed algorithm assures a very long lifetime with respect to PF scheme. We have an abrupt decreasing in lifetime for $N = N_{crit}$ due to the additional work of compression.

As seen in Section VI, the algorithm depends also on some others technical parameters such as size of the scalar data read by sensor and dimension in bytes of the sensor ID.

An increasing in the ID field of the payload does not consistently affect the performance of the algorithm since it does not depend heavily on B_{ID} as seen in equation (8). The increase of the number of bytes used for ID has the main effect to enlarge the central region for DCS since PF performs very bad when it has to manage too many bytes per node. In Fig. 10 it is clear the trend of lifetime when parameter B_{ID} is changed.

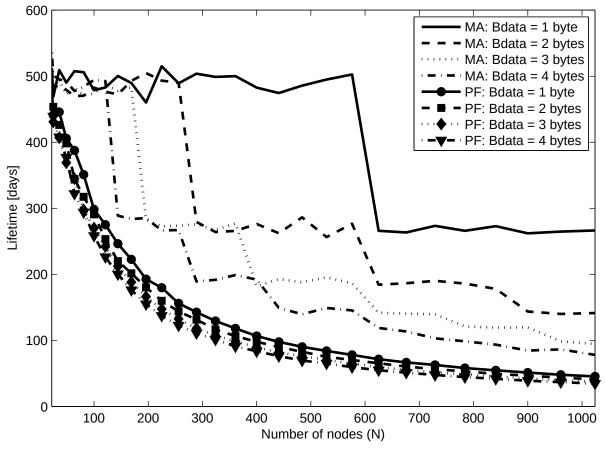

Fig. 11: Lifetime of the network: comparison between PF and Mixed Algorithm (MA) varying B_{data}.

A bit less intuitive is the behavior of the WSN when B_{data} is modified. Both PF and DCS depends on B_{data}. Fig. 11 illustrates how the performance of DCS decreases for bigger values of the parameter even if the lifetime continues to be longer with respect to the classical gathering scheme.

The last parameter we address is the extra-payload, added by sporadic readings as described in subsection VI-A. To work with the worst case, in our simulation we assume that these sporadic readings start from the leaves of the network more distant from the sink, so that the additional payload is forwarded along all the routing path traversing as many nodes as possible. Since in general these packets are not very frequent, their impact on lifetime of the network is very limited. Nevertheless they stress the potency of the algorithm: they can change run-time the behavior of the nodes these packets traverse. Fig. 12 shows the percentage of the nodes in which is convenient perform compression with DCS in two different

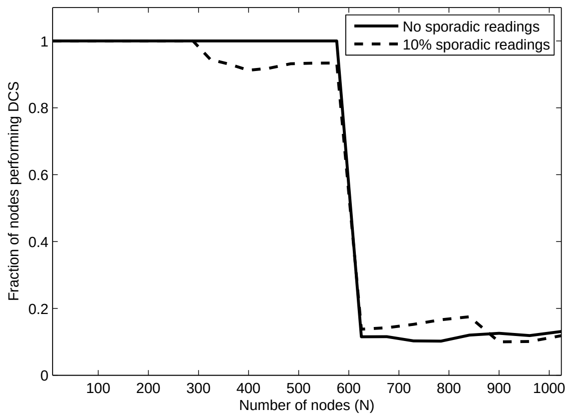

Fig. 12: Fraction of the nodes performing DCS with 0% and 10% of the nodes rising sporadic readings.

case: none of the nodes rise sporadic readings and 10% of the nodes perform sporadic reading with $B_{incomp} = 60$ bytes. These two cases can be seen as two different moments in the same network: without a central coordination, the network self-organizes to reduce the number of packets circulating. The plot presents an fluctuating trend in function of the network size. This is due the difference between the size of incompressible data and the free space inside the payload of the ZigBee packet before a new packet is needed to incapsulate the whole data. If B_{incomp} bytes fit inside the last packet to send, the extra-payload is sent ”for free” from the point of view of the number of packets. Otherwise this additional payload determines the sending of a new packet.

VIII. Conclusions

In this paper we have presented a new kind of distributed algorithm for optimization of CS when ZigBee protocol is used for communication. This new algorithm has been compared to a classical gathering scheme named *Pack & Forward* and to classical DCS. We have shown how, with this mixed algorithm, the performances and the lifetime of the system can be increased in a fully automated and distributed way. We have also analyzed the behavior of the solution when some parameters about sensor readings are changed, showing that our solution keeps to be better. Furthermore we have addressed the problem of sporadic readings and demonstrated that the network reacts to this change without need for a central coordinator.

There are several promising avenue of research that we can explore. We have not considered other type of routing algorithm apart from geographical routing or temporal correlation and temporal compression that can provide our scheme better results in terms of in-network traffic.

References

[1] S.S. Pradhan, J. Kusuma and K. Ramchandran ”Distributed Compression in a Dense Microsensor Network”, *IEEE Signal Processing Magazine*, 2002.

[2] S. Pattem, B. Krishnamachari and R. Govidan ”The Impact of Spatial Correlation on Routing with Compression in Wireless Sensor Networks”, *Information Processing in Sensor Network (IPSN04)*, 2004.

[3] D. Baron, M.F. Duarte, S. Sarvotham, M.B. Wakin and R.G. Baraniuk ”An Information-Theoretic Approach to Distributed Compressed Sensing”, *Allerton Conference on Communication, Control, and Computing*, 2005.

[4] G. Quer, R. Masiero, D. Munaretto, M. Rossi, J. Widmer and M. Zorzi ”On the Interplay Between Routing and Signal Representation for Compressive Sensing in Wireless Sensor Networks”, *Third International Conference on Internet Technologies and Applications (ITA09)*, 2009.

[5] S. Lee, S. Pattem, M. Sathiamoorthy, B. Krishnamachari and A. Ortega ”Compressed Sensing and Routing in Multi-Hop Networks”, *Technical Report, University of Southern California*, 2009.

[6] W. Wang, M. Garofalakis and K. Ramchandran ”Distributed Sparse Random Projections for Refinable Approximation”, *Information Processing in Sensor Network (IPSN07)*, 2007.

[7] C. Luo, F. Wu, J. Sun and C. Wen Chen ”Compressive Data Gathering for Large-Scale Wireless Sensor Networks”, *MobiCom’09*, 2009.

[8] ”LUCE - SensorScope Wireless Distributed Sensing System for Environmental Monitoring”. Available: http://sensorscope.epfl.ch/index.php/LUCE

[9] J. Haupt, W.U. Bajwa, M. Rabbat and R. Nowak ”Compressed Sensing for Networked Data”, *IEEE Signal Processing Magazine*, March, 2009.

[10] D. Slepian and J.K. Wolf ”Noiseless Coding of Correlated Information Sources”, *IEEE Trans. Inform. Theory*, vol. IT-19, pp.471-480, July 1973.

[11] E.J. Candes and M. Wakin ”An Introduction To Compressive Sampling”, *IEEE Signal Processing Magazine*, March 2008.

[12] E.J. Candes ”Compressive Sampling”, *Proceedings of the International Congress of Mathematicians*, Madrid, Spain, 2006

[13] J. Stanford and S. Tongngam ”Approximation Algorithm for Maximum Lifetime in Wireless Sensor Networks with Data Aggregation”, *International Journal of Sensor Networks*, vol. 6-1, pp.44-50, 2009.

[14] R. Rajagopalan and P.K. Varshney ”Data aggregation techniques in sensor networks: A survey”, *IEEE Communications Surveys & Tutorials*, 2006.

[15] C.M. Sadler and M. Martonosi ”Data compression algorithms for energy-constrained devices in delay tolerant networks”, *Conference On Embedded Networked Sensor Systems*, 2006.

[16] F. Marcelloni and M. Vecchio ”A Simple Algorithm for Data Compression in Wireless Sensor Networks”, *IEEE communications letters*, June 2008.

[17] ”Ember EM250 datasheet”. Available: http://www.ember.com/pdf/EM250_Datasheet.pdf