Reliable Data Delivery in sparse WSNs with Multiple Mobile Sinks: an Experimental Analysis

Giuseppe Anastasi*, Eleonora Borgia†, Marco Conti†, and Mario Di Francesco‡

*Dept. of Information Engineering
University of Pisa (Italy)
E-mail: g.anastasi@iet.unipi.it

†Institute of Informatics and Telematics
National Research Council (Italy)
E-mail: {eleonora.borgia, marco.conti,}@iit.cnr.it

‡CReWMaN
University of Texas at Arlington
E-mail: mariodf@uta.edu

***Abstract*—Urban sensing is emerging as a significant Wireless Sensor Networks (WSNs) application. In such a scenario, static sensors are sparsely deployed in an urban area to collect environmental information. Sensed data are opportunistically collected by Mobile Sinks (MSs), which can be other sensor nodes attached to cars or buses, or carried by people while they move around the city. Since the contacts between the MSs and the static sensors are infrequent and short, reliable and energy efficient data collection is a primary concern of such applications. To this end, we exploit a hybrid data delivery scheme based on both Erasure Coding (EC) and feedback by the MSs. We provide an optimized implementation, and show by extensive experiments in a real testbed that the proposed approach is feasible, despite the very limited storage and processing resources of commercially available sensor platforms.**

***Keywords*-Urban sensing, wireless sensor networks, mobile sinks, reliable data delivery, erasure coding, implementation.**

I. Introduction

Sparse WSNs are particularly appealing for urban applications [1]. In this case, a low number of static sensors can be placed in a few key points of a city to collect environmental data, e.g., the level of pollutants or allergens in the air. Data generated by sensors can be collected by Mobile Elements (ME), which can be either vehicles (e.g., cars, buses or shuttles) or people carrying sensing devices or smartphones. As a result, the network lifetime can be significantly improved [2]. MEs can either use the collected information for their own purposes, or make them available to remote users through a high range Internet connection. Since in both cases they are the endpoints of data collection in the WSNs, they act as Mobile Sinks (MSs) [3].

In such a scenario, data collection presents several issues. One of them is the limited contact time between the sensors and the MSs. This is especially true when MSs are represented by vehicles. A different aspect is related to the high and variable error probability of wireless communications. The wireless channel is known to be noisy, especially in urban scenarios where there might be several sources of interferences. Since the sink is moving, the message loss rate is highly time-varying, and significantly affected by the physical distance between the sensors and the MSs [1]. As a consequence, communication protocols targeted to data delivery in sparse WSNs with MSs should be reliable, and should also have a limited overhead, in order to exploit the short and scarce contacts to the full extent. In this context, approaches based on on Automatic Repeat reQuest (ARQ) schemes have several limitations, including a high overhead, and are not suitable for broadcast communications [4, 5]. On the contrary, approaches based on Erasure Coding (EC) have shown to be effective [6].

To deal with all the issues above, in [7] we proposed an adaptive communication protocol for reliable data delivery in sparse sensor networks with multiple MSs, which efficiently combines EC with an ARQ scheme. The simulation analysis carried out in [7] has shown that the proposed hybrid communication protocol outperforms a pure ARQ scheme based on acknowledgements and selective retransmissions, especially when there are many MSs simultaneously in contact with the sensor. However, such analysis is based on simulations and does not consider the effect of resource limitations imposed by real sensor devices. For example, in [7] we considered the maximum *stretch factor* (i.e., ratio between the number of redundant and original messages) in order to investigate the potentials of EC, but this may be unfeasible in many real sensor platforms due to memory limitations. Thus, the major contribution of this paper is to analyze the real effectiveness of the proposed approach on a real sensor network platform with limited computation and storage resources. To this aim, in this work we focus our analysis on resource utilization. First, we investigate resource utilization in terms of used memory, as the amount of RAM represents the major limiting factor for the EC scheme. We also evaluate the computational burden for encoding and decoding data, since they are strictly connected with the specific EC parameters. Another major contribution is the evaluation of the additional energy consumption due to the encoding phase, which obviously depends on the specific sensor platform which is considered. Experiments in our real testbed show that our approach is feasible, despite the very limited storage and processing resources of commercially available sensor platforms, and suitable to urban scenarios where more than one MS can be in contact with a static sensor at once.

The rest of the paper is organized as follows. Section II

This work has been partially supported by NSF Grants CNS-1049652 and CNS-1050618.

presents the motivation and the system model. Section III describes the proposed reliable data delivery scheme. Section IV presents the experimental evaluation of our protocol. Finally, Section V provides the experimental results.

II. Motivation and system model

In this paper we will limit our attention to a specific class of WSN applications, referred to as *bundle-oriented applications*. In this context, static sensors produce a limited amount of data. For instance, such data might consist in a detailed snapshot of some quantity of interest (e.g., the level of pollutants in the air) collected during a time-frame (e.g., the current day or the last few hours). The data stored at the sensors are opportunistically delivered to mobile users whenever they are in contact. Specifically, the data transfer is accomplished by streaming the messages in the bundle to the mobile users until all data have been successfully transferred. Mobile users consume data on their own, i.e., they are the final endpoint of the data transfer. Therefore, they behave as *Mobile Sinks* (MSs). While static sensors are resource-constrained, especially in terms of energy, MSs have higher computational resources and no major energy concerns (as their battery can be recharged). A realistic application scenario is represented by sensors located in an urban environment (e.g., along streets, at traffic lights, at bus stops) where MSs are represented by people walking or cars moving around the city.

In such a scenario, contacts occur infrequently and only for a short time. Hence, they should be exploited as much as possible by the communication protocol in order to efficiently deliver data. In detail, the duration of contacts (i.e., the *contact time*) is very limited, especially when the speed of the MSs is high, or the sensor nodes use a power management scheme – e.g., a duty-cycle mechanism [1] – to save energy. In addition, data transfer might be affected by a severe message loss, due to the distance and interferences. This is especially true for scenarios where a large number of elements (including MSs) are simultaneously present in the communication range of the sensors.

According to [8], scenarios with short and unpredictable contacts benefit from communication protocols which do not require frequent feedback nor synchronization. In this context, communication protocols exploiting *Erasure Coding* (EC) [9] are particularly appealing, especially for broadcast transmissions [10]. Essentially, EC schemes split source data into a number of blocks, which are then codified into a redundant set which still represent the original data. The advantage is that original data can be correctly reconstructed even though part of encoded data is lost. EC schemes specifically designed for communication protocols map data blocks to messages which are then transmitted over the radio. In detail, a source node willing to send m messages encodes the m messages into r redundant messages, with $r \gg m$. A destination node does not need to receive the exact m original messages; indeed, any set of m messages out of the r encoded messages is enough to decode the m original messages. This property significantly improves the robustness of the system against data loss.

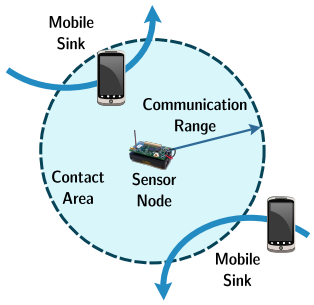


Figure 1. Reference network model.

A. System model

The reference network model is illustrated in Figure 1. We assume that the network is sparse enough so that static sensors cannot communicate with each other, but can only communicate with one or more MSs, (i.e., more than one MS can be in the communication range of the static node at the same time). The contacts are unpredictable, and the MSs are independent, i.e., there is no coordination between them.

Communication between the sensor nodes and the MSs is possible only when they are in contact, i.e., in the same communication range. As the mobility pattern of the MSs is assumed to be random, the static sensor has to: *i)* discover the arrival of a MS in the contact area; *ii)* detect when a MS has left the contact area, i.e., it is not reachable any more. As for the first aspect, low-power discovery protocols can be used [2], like those based on periodic listening [11]. The second aspect is related to the unknown duration of contacts. Specifically, once a contact has started, the sensor cannot derive if a MS is still reachable or not at a given time, unless the MS provides explicit feedback on its presence in the contact area. In the following we assume that MSs periodically signal their presence in the contact area by sending special control messages to the static sensors.

We exploit the feedback messages to complement a communication scheme based on EC with acknowledgements and retransmissions. The resulting approach is *hybrid* since it combines the different (proactive and reactive) schemes for reliable data delivery. As shown in [12], hybrid schemes can increase the *effective transmission range* of a node, defined as the area where messages are delivered within a given error probability. This is especially useful for sparse WSNs where nodes have a short contact time with MSs. The details of the reliable data delivery scheme are given in the next section.

III. Reliable data delivery

In this section we will discuss the different phases of the proposed data delivery scheme – namely, *encoding*, *communication*, and *decoding* – with focus on the resource constraints typical of sensor nodes.

A. Encoding

The *encoding* phase is executed at the static sensor node, and consists in adding redundancy to the source bundle by applying some EC technique. There are two basic options for EC. The first consists in using conventional erasure codes, such as the Reed-Solomon (RS) codes [13]. The other case is given by rateless codes (also called *fountain codes*) [10], that can be applied to a sequence of source data which is potentially unlimited. However, fountain codes are asymptotically optimal, i.e., they ensure that any m out of the r source elements are needed to recover the original message only when m is very large. Actually, the size of the source bundle is rather limited in the considered scenario, also due to the limited memory resources of sensor nodes. On the other hand, RS codes are optimal for the considered scenario, even though they have a higher computational cost. Indeed, optimized versions have been proposed in the literature [14], and it has been shown that implementations are feasible even in sensor node platforms [6].

In the following we will use the RS codes as the EC scheme, and incorporate the main optimizations introduced in [14] to reduce the computational complexity: computations in finite fields, lookup tables, and systematic codes. Specifically, *systematic codes* are such that the coded data include a verbatim copy of the source elements. Therefore, m out of the r elements do not even need to be encoded; hence, systematic codes reduce the computational complexity, and can also help to reduce the memory usage as a side effect.

In order to keep m small and independent from the bundle size, the source data are split into B blocks (i.e., $b_0, b_1, \ldots, b_B - 1$), each consisting of m data units, similarly to [14]. Each block is then encoded separately to produce r data units. The ratio between the number of redundant and original messages is named *stretch factor* (i.e., $s_f \triangleq r/m$). Encoding is performed by the sensor node in advance, whenever the source data are ready. Hence, the sensor node can initiate the communication with a MS as soon as its presence is detected, without having to encode data units on the fly. As a result, the utilization of the (limited) contact time is increased, since no additional encoding overhead occurs during communication.

B. Communication

The *communication* phase starts when the sensor node has detected the presence of one or more MSs in the contact area. We assume that both the sensor nodes and the MSs are aware of the encoding parameters, i.e., the number of original messages (m) and blocks (B) within a bundle, as well as the encoding function.

Reliable communication is accomplished by means of the *Hybrid Adaptive Interleaved Communication Protocol* (HI) [7]. Figure 2 shows the communication procedure initiated at the sensor side upon discovering the presence of at least

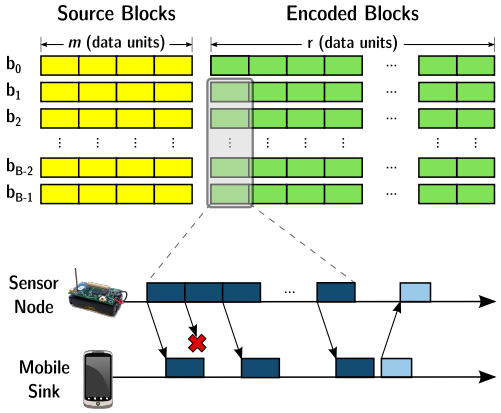


Figure 2. Communication phase.

one MS. To transmit encoded data units, the sensor node uses an interleaved scheme which consists in scheduling encoded data units picked from distinct (consecutive) blocks rather than sequentially from the same block. This procedure guarantees more uniform message losses among all blocks, and is independent from the number of blocks and the bundle size. Then the sensor node encapsulates one encoded data unit into a message of size b_{msg} bytes, and transmits a burst of messages to the MSs. The MS stores the encoded data units from messages correctly received into its local buffer. In addition, the MS uses the *block identifier* and the *sequence number within the block*[1] from the encoded message header to derive if a sufficient number of messages (i.e., at least m different messages for each block) has been received to decode the original bundle.

Every T_{ack} time, the MS replies with an acknowledgement message[2], which also provides a feedback of the MS presence in the contact area. More specifically, a MS is assumed to be out of contact when no acknowledgements are received within an *end of contact* time frame T_{eoc}. Acknowledgements also carry a mask which notifies, for each block, how many encoded messages have been correctly received. The sensor node collects all the incoming acknowledgements from all MSs that are in contact, and stores, for each block, the lowest value of messages received correctly by all the MSs. From the quantities above, the sensor is able to derive whether additional data transmissions are required or not. Specifically, the sensor transmits additional encoded messages for all the blocks for which less than m messages have been received. In order to transmit always fresh and useful encoded messages, the sensor starts from the last message sent but skipping those blocks already completed by all MSs (if any). The process is repeated until the minimum set of encoded messages has been received by all the MSs (i.e., all the block values stored at the sensor node are equal to m), or all MSs are out of the contact area.

[1] Allowed block identifier values are in the range $[0, B-1]$, while allowed the sequence numbers within the block are in the range $[1, r]$.

[2] A contention-based approach is used to avoid collisions between multiple MSs.

Table I
PARAMETERS USED FOR THE EVALUATION

Parameter	Value
Message (payload) size (b_{msg})	89 bytes
Frame size	128 bytes
Total message transmission time (T_{msg})	17 ms
Acknowledgement period (T_{ack})	$16 \cdot T_{msg}$
Beacon period	100 ms
End of contact timeout (T_{eoc})	$8 \cdot T_{ack}$
Duty-cycle	5%
MS speed	40 km/h
Contact time	17 s
Transmission power at 0 dBm (P_{tx})	52.2 mW
Receive power (P_{rx})	56.4 mW
PCU power when the radio is off (P_e)	5.4 mW

It is worthwhile noting that the protocol is able to dynamically adapt to different levels of message losses experienced by different MSs. In addition, the acknowledgements introduce a very limited overhead, as they are anyway needed as explicit feedback on the MS presence in the contact area.

C. Decoding

The *decoding* phase is performed at the receiver side (i.e., at the MS) when m distinct encoded messages have been received for each block. The MS decodes the messages and stores the resulting block in its local buffer. Once all B blocks have been correctly decoded, the MS obtains a copy of the original bundle which can be used by the application. On the contrary, if all the required encoded messages are not received by the MS, the decoding cannot be performed, and an error message is reported to the application. Note that, for data reconstruction, we adopt similar software optimizations to those used in the encoding process.

IV. EXPERIMENTAL EVALUATION

Before presenting the experimental results, we will briefly describe the experimental testbed, the used methodology, and the performance metrics considered in the evaluation.

A. Experimental setup

We performed our experiments in a testbed of Tmote Sky sensors [15], whose major components are: *i)* a Texas Instruments MSP430 microcontroller running at a 8 MHz clock, and equipped with 10 KB RAM and 48 KB program memory, and *ii)* an IEEE 802.15.4-compliant Chipcon Wireless Transceiver, capable of a raw bitrate of 250 kbps over the unlicensed 2.4 GHz frequency band. The Tmote Sky is also supported by the TinyOS operating system [16], which we used for implementing the reliable data delivery scheme.

Our experimental scenario is represented by a fixed sensor node and a variable number of MSs (from 1 to 5). In this scenario, the MSs approach the sensor by moving at a constant speed of 40 km/h on a linear path at a fixed (vertical) distance from the sensor, and enter the contact area randomly. Due to the high variability of channel conditions, obtaining comparable experiments (i.e., with the same statistics) is quite difficult. Thus, to guarantee repeatable experiments we emulated the message loss due to mobility according to the realistic model in [1] expressed as a 2-degree polynomial function. The obtained contact time between the sensor and the MSs is 17 s.

In each experiment, a bundle of given size is first encoded and then sent by the static sensor to the MSs in the contact area. Each experiment is replicated 100 times, and the results are averaged over all replicas (the standard deviations are also provided as error bars). The parameters used in the evaluation are summarized in Table I.

B. Performance metrics

As for the resource utilization, we will consider the following metrics:

- *Memory usage*: the total amount of RAM, expressed as percentage with respect to the total available RAM (i.e., 10 KB), required by the reliable data delivery scheme both for the encoding and communication;
- *Encoding time*: the amount of time needed to encode the bundle, measured at the sensor node;
- *Decoding time*: the amount of time needed to decode the bundle[3], measured at the MS side;
- *Encoding energy*: the average energy consumed by a sensor node per each useful byte.

Specifically, the encoding energy is obtained as:

$$E_e = s_f \cdot \frac{T_{code} \cdot P_e}{b_{msg}} \quad (1)$$

where $s_f \triangleq r/m$; T_{code} is average time required to produce an encoded data unit; P_e is the power consumed by the sensor during the encoding phase (i.e., the energy consumed without considering the communication costs); finally, b_{msg} is the size (in bytes) of the message payload.

V. EXPERIMENTAL RESULTS

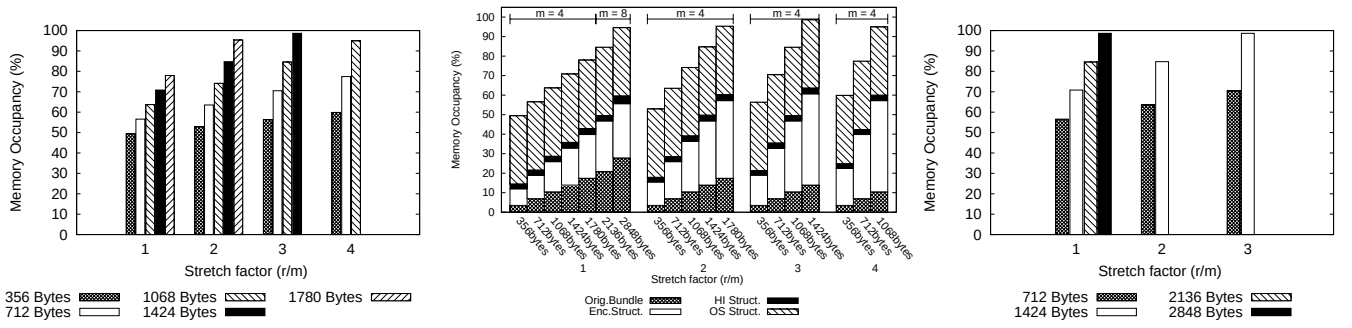
In this section we will evaluate the performance of the data-delivery protocol described in Section III.

We start our analysis by considering the resource utilization in terms of the used memory, since the amount of RAM represents the major limiting factor for both the bundle size and the encoding parameters. Then, we investigate the time needed to encode (decode) each bundle, as well as the energy consumed during the encoding process, since they strictly depend on the specific encoding parameters. All the quantities above are evaluated for different bundle sizes and for different levels of redundancy. Specifically, we consider $m = 4$ and $m = 8$ and s_f in the range[4] [1,4].

The percentages of memory utilization at the sensor node for $m = 4$ are illustrated in Figure 3a. As expected, the

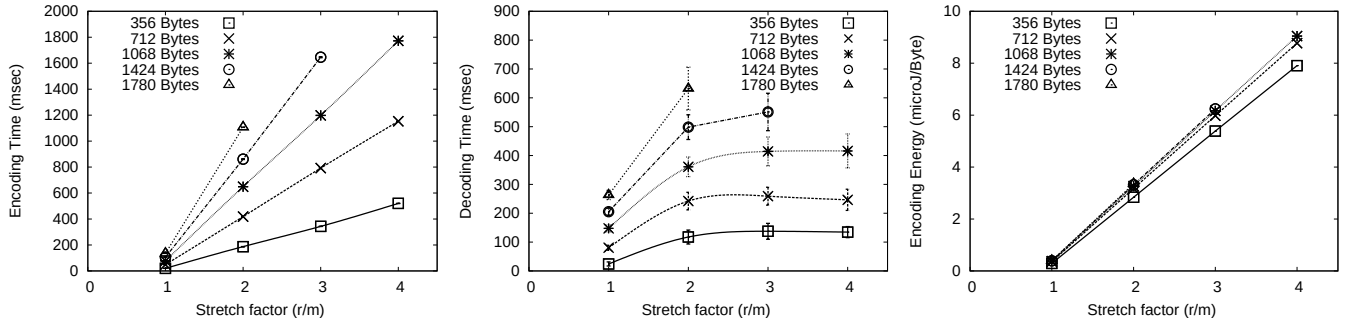
[3]The decoding time is computed only for the MSs which have correctly received the entire bundle. For comparison purposes, we assume that the MSs is using the same hardware platform and the software implementation of the reliable data delivery scheme as the static sensor.

[4]Note that $s_f = 1$ means that no codes are produced but the sensor transmits the original bundle.



(a) (b) (c)

Figure 3. Memory usage for (a) $m = 4$ and (c) $m = 8$. (b) Breakdown of the memory usage for different bundle size.



(a) (b) (c)

Figure 4. (a) Encoding time, (b) decoding time, and (c) encoding energy as a function of the stretch factor for different bundle sizes where $m = 4$.

memory usage depends on the size of the bundle and of the level of redundancy introduced. The higher the bundle size and the stretch factor, the higher is the memory usage. For instance, encoding the smallest bundle (i.e., 356 bytes which represents about 3.5% of the RAM) with $s_f = 1$ consumes half of the available RAM in the system. On the other side, encoding 1424 bytes (representing about 13.5% of the RAM) with $s_f = 3$ almost uses all the memory. As a result, when using $s_f = 4$ the maximum bundle size that can be encoded is approximately 1 KB. This large increase in the memory usage can be explained by taking into account how the available sensor node memory is used by the different components of the data delivery protocol. Figure 3b breaks down the memory usage according to four different factors. Specifically, the used RAM depends on the buffer size for storing the original bundle and the encoding structures (i.e., the encoding matrix, the lookup tables, and the encoded bundle). Additional data structures are used by the communication protocol. Finally, part of the RAM is used by the variables and the data structures required by the operating system (e.g., the queues used for data transmission and reception, timers). As highlighted by the figure, most of the available RAM (i.e., 10 KB) is used by the operating system structures (i.e., *OS Struct.* in the figure), which account for a 35% share of the RAM, irrespective of the encoding parameters and the bundle size. A different contribution, which is constant, is represented by data structures of the communication protocol (i.e., *HI Struct.* in the figure). However, this factor has a very small impact on memory usage (i.e., less than 3%). In contrast with the two factors already discussed, the encoding contribution is variable and strictly dependent on the bundle size and stretch factor. It is up to 20% of the total RAM size for the smallest bundle, while it exceeds 40% for the following (*bundle size, stretch factor*) pairs: (1780, 2), (1424, 3), (1068, 4). In all the other cases, the encoding contribution falls in the range (20%, 35%). More specifically, for a given bundle size and stretch factor, the memory usage by the encoding process can be derived as:

$$M_{tot} = \alpha + s_f \cdot b_{tot} \quad (2)$$

where α is a constant value representing the total amount of bytes required for the encoding matrix, the lookup tables, and a set of variables. This confirms the linear growth of the encoding contribution.

We also considered the case $m = 8$ (see Figure 3c). The increase in the number of source data units results in a decrease for both the maximum bundle size and the stretch factor that can be used by the sensor. Figure 3c highlights that it is possible to increase the stretch factor up to 3 for bundle sizes smaller than 1.5 KB; for higher bundle sizes, the maximum stretch factor is limited to 1, as more than 60% of the RAM is used for all the data structures required by the reliable data delivery scheme.

Figure 4a shows the encoding time for $m = 4$. As depicted by the figure, the encoding time increases linearly with the stretch factor. Note that the encoding process takes

Table II
ENCODING TIME, DECODING TIME, AND ENCODING ENERGY AS A FUNCTION OF THE STRETCH FACTOR FOR $m = 8$

Bundle size	**Encoding time (ms)**			**Decoding time (ms)**			**Encoding energy (μJ/bytes)**		
(bytes)	$s_f = 1$	$s_f = 2$	$s_f = 3$	$s_f = 1$	$s_f = 2$	$s_f = 3$	$s_f = 1$	$s_f = 2$	$s_f = 3$
712	41.2805	718.652	1395.08	66.6473	399.361	411.859	0.302474	5.28353	10.2673
1424	99.9418	1608.92	2718.64	193.813	846.286	1068.12	0.375482	5.93352	8.66746
2136	163.219	n.a.	n.a.	332.427	n.a.	n.a.	0.399871	n.a.	n.a.
2848	223.873	n.a.	n.a.	432.233	n.a.	n.a.	0.411727	n.a.	n.a.

less time when the stretch factor is equal to 1, requiring less than 200 ms for the largest bundle. This is due to the use of systematic codes in the encoding process. This means that the first m data units are just copied in memory, requiring about 6 ms each. On the contrary, the production of additional data units takes more time due to the encoding operations such as the matrix-by-vector multiplication. Specifically, we measured about 42 ms, on average, to generate each additional code (i.e., one order of magnitude higher than the simple memory copy). In addition it is worth pointing out that, since the time for generating encoded data units is higher than the total transmission time of a packet (17 ms), it is not possible to encode data units on the fly. This confirms the effectiveness of our choice, where the entire bundle is encoded in advance, without consuming the limited contact time.

Figure 4b shows the decoding time for $m = 4$, which is linear for stretch factors up to 2, and almost constant later. Since systematic codes are used, the decoding time strongly depends on how many (copies of the) original messages have been received at the MS side. We have found that in the considered scenario, a significant percentage (i.e., 40% -50%) of the received encoded messages consists in a copy of the original data unit. Overall, the decoding phase remains in the order of hundreds of milliseconds – about 100 ms for the smallest bundle, and about 700 ms for the largest one. Hence, the decoding delay is negligible if compared with the time needed to actually transfer the bundle (i.e., up to 3 sec for the largest bundle). The results also confirm that, as expected, the decoding phase is faster than the encoding phase, requiring about half of the encoding time.

Figure 4c represents the average energy spent to encode the bundle. The trend of the curve is linear, according to Equation (1). The curves are almost overlapped, with a slight difference only for values of s_f higher than 3. What is important to highlight here is that the energy consumption for encoding bytes is very limited and low and requires only few micro Joules. This is fundamental in order to save energy at the sensor side.

Finally, Table II provides the encoding time, the decoding time, and the energy consumed for encoding the bundle when $m = 8$. As shown in the table, they are aligned with the results obtained for the case $m = 4$. The encoding time is limited to a few milliseconds in the case of small bundles, while it is of 2.7 s for the 1424 bytes bundle and $s_f = 3$. The decoding time is approximately one half than the encoding time, and the energy spent to encode the bundle is still equal to a few micro Joules.

To conclude the analysis, we have also evaluated the reliability and the efficiency of the communication protocol. Results of such analysis, omitted here for sake of space, have demonstrated that our approach can achieve a high probability of correct data delivery, as well as a high energy efficiency, especially when multiple MSs are in contact with static sensors at the same time. Interested readers may refer to [17] for the details.

REFERENCES

[1] G. Anastasi, M. Conti, E. Gregori, C. Spagoni, and G. Valente, "Motes sensor networks in dynamic scenarios," *Int. Jour. of Ubiquitous Computing and Intelligence*, vol. 1, no. 1, April 2007.

[2] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad Hoc Networks*, vol. 7, no. 3, May 2009.

[3] M. Di Francesco, S. K. Das, and G. Anastasi, "Data collection in wireless sensor networks with mobile elements: A survey," *ACM Trans. Sensor Networks*, to appear.

[4] A. Somasundara, A. Kansal, D. Jea, D. Estrin, and M. Srivastava, "Controllably mobile infrastructure for low energy embedded networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 8, 2006.

[5] A. Kansal, A. Somasundara, D. Jea, M. Srivastava, and D. Estrin, "Intelligent fluid infrastructure for embedded networks," in *Proc. ACM Mobisys 2004*, 2004.

[6] S. Kim, R. Fonseca, and D. Culler, "Reliable transfer on wireless sensor networks," in *Proc. of IEEE SECON 2004*, 2004.

[7] G. Anastasi, E. Borgia, M. Conti, and E. Gregori, "A hybrid adaptive protocol for reliable data delivery in wsns with multipe mobile sinks," *The Computer Journal*, vol. 54, no. 2, pp. 213–229, February 2011.

[8] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Rfc 4838 delay-tolerant networking architecture," http://tools.ietf.org/html/4838, 2007.

[9] S. Lin and D. Costello, *Error Control Coding*, 2nd ed. Prentice Hall, 2004.

[10] J. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE Jour. on Sel. Areas in Communications*, vol. 20, no. 8, pp. 1528–1540, Oct. 2002.

[11] G. Anastasi, M. Conti, and M. Di Francesco, "Reliable and energy-efficient data collection in sparse sensor networks with mobile elements," *Elsevier Performance Evaluation*, vol. 66, no. 12, pp. 791–810, December 2009.

[12] M. C. Vuran and I. F. Akyildiz, "Error control in wireless sensor networks: a cross layer analysis," *IEEE/ACM Trans. Netw.*, vol. 17, pp. 1186–1199, August 2009.

[13] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journ. Soc. for Ind. and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[14] L. Rizzo and L. Vicisano, "Rmdp: an fec-based reliable multicast protocol for wireless environments," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 2, pp. 23–31, April 1998.

[15] Sentilla Corporation, "Tmote Sky Datasheet v.1.04," http://sentilla.com/files/pdf/eol/tmote-sky-datasheet.pdf.

[16] "TinyOS," http://www.tinyos.net/.

[17] G. Anastasi, E. Borgia, M. Conti, and M. Di Francesco, "Reliable data delivery in sparse WSNs with multipe mobile sinks: an experimental analysis," IIT-CNR, Tech. Rep. 5-2011, April 2011, http://www.iit.cnr.it/node/9133.