Maximizing Growth Codes Utility in Large-Scale Wireless Sensor Networks

Yao Zhao, Xin Wang, Jin Zhao, and Xiangyang Xue

School of Computer Science, Fudan University, China
Shanghai Key Lab of Intelligent Information Processing, Shanghai, China

Abstract. The goal of Growth Codes proposed by Karma et.al. is to increase the “persistence” of sensed data, so as to promise that data is more likely to reach a data sink. In many “zero-configuration” sensor networks, where the network topology would change very rapidly, Growth Codes are especially useful. However, the design of Growth Codes is based on two assumptions: (1) each sensor node contains only one single-snapshot of the monitored environment, and each packet contains only one sensed symbol; (2) all codewords have the same probability to be received by the sink. Obviously, these two assumptions do not hold in many practical scenarios of large-scale sensor networks, thus the performance of Growth Codes would be sub-optimal. In this paper, we generalize the scenarios to include multi-snapshot and less random encounters. By associating the decimal degree with the codewords, and by using priority broadcast to exchange codewords, we aim to achieve a better performance of Growth Codes over a wider range of sensor networks applications. The proposed approaches are described in detail by means of both analysis and simulations.

1 Introduction

Wireless sensor networks have been widely used for data perception and collection in different scenarios, such as floods, fires, earthquakes, and military areas. Often in such networks, the sensor nodes used to collect and deliver data are prone to catch failure suddenly and unpredictably themselves. Thus, all the protocols designed for these sensor networks should focus on the reliability of data collection and temporarily store part of the data for information survives. Many coding approaches are proposed to achieving the robustness in such networks, since coding over the sensed data may increase the likelihood that the data will survive as some nodes fail. Among them, Growth Codes [7] are specially designed for the purpose of increasing the “persistence” of the sensed data. Assuming that the network topology changes very rapidly, *e.g.*, due to link instability or node failures, Growth Codes address to increase the *data persistence*, where data persistence is defined as the fraction of data generated within the network and eventually reaches the sink(s). The codewords employ a dynamically changing codeword degree distribution to deliver data at a much faster rate to the sink and promise to be able to decode a substantial number of the codewords at any given time.

However, the design of Growth Codes is based on two assumptions: (1) each sensor node contains only one single-snapshot of the monitored environment, and each packet contains only one sensed symbol; (2) all codewords have the same probability to be received by the sink. This is true for information exchange at random encounters among nodes or for very high mobility. However, it does not well capture protocol behavior in realistic large-scale sensor networks, where the performance of Growth Codes would be sub-optimal. In this paper, we loose the above assumptions and generalize the scenarios of original Growth Codes to a wider range of applications.

Firstly, the design of Growth Codes is generalized to the scenarios of multi-snapshot, where the buffer of each sensor node can store multiple symbols sensed from the monitored area, and the size of a transmission packet is larger than the size of a sensed symbol. Notice that, this assumption is quite reasonable in practical applications of large-scale sensor network, *e.g.*, a transmission packet may contain more than hundreds of bytes, whereas a sensed symbol may contain only several bytes. For clarity to describe, we term “*packet*” as the transmission data unit and term “*symbol*” as the sensed data unit, respectively. Thus, when a node receives an input packet, it can disassemble this packet into several symbols and then generate a new packet over other symbols in its local buffer. In this case, Growth Codes can be modified to encode on the symbol level, instead of on the packet level, to achieve a better utility in the multi-snapshot scenarios.

Secondly, the design of Growth Codes is generalized to the scenarios of less random encounters, where the network scale is too large to promise that the all codewords could have the same probability to reach to the sink. Often in such scenarios, the nodes located far from the sink would have less chance to deliver their symbols to the sink than the closer ones. In this case, we need investigate special technique to increase the chance of the symbols which are sensed in an area far from the sink. Motivated by the natural property of broadcasting in wireless sensor networks, we introduce priority broadcast to disseminate sensed data, which gives a high priority to the farther data. By priority broadcast, we show that even in the scenarios of less random encounters, the performance of Growth Codes can approach to the theoretical value.

The rest of the paper is organized as follows. Section 2 gives an overview of the related works. Section 3 gives a brief overview of data persistence in large scale wireless sensor networks and introduces the original Growth Codes. In Section 4, the design of Growth Codes is modified to encode on the symbol level to maximize utility in the case of multi-snapshot. In Section 5, the design of Growth Codes is modified to use priority broadcast to maximize utility in the case of large network scale. Section 6 evaluates the performance of our proposed modifications by simulations. Section 7 concludes this paper.

2 Related Works

Storing and disseminating network coded [2] information instead of the original data can bring significant performance improvements to wireless sensor network

protocols [1]. However, though network coding is generally beneficial, coding over all available packets might leave symbols undecodable until the sink receives enough codewords.

A decentralized implementation of erasure codes is proposed in [5]. Assuming that there are n storage nodes with limited memory and $k < n$ sources generating the data, the authors consider that the sink is able to retrieve all the data by query any k nodes. Similarly, decentralized fountain codes [8] [3] are proposed to persist the cached data in wireless sensor networks.

Growth Codes [7] are specifically designed to increase the “persistence” of the sensed data in dynamic environments. In Growth Codes, nodes initially disseminate the original data and then gradually code over other data as well to increase the probability that information survives. To achieve more data persistence under scenarios of different node mobility, resilient coding algorithms [10] are proposed. The authors propose to keep a balance between coding over all incoming packets and Growth Codes. Furthermore, the approach of Growth Codes is generalized to the multi-snapshots scenarios in [9], where the authors aim to maximize the expected utility gain through joint coding and scheduling and propose two algorithms, with and without mixing different snapshots. When dealing with multi-snapshot, the authors prove the existence of the optimal codeword degree but do not propose any specific methods.

Border node retransmission based probabilistic broadcast, is proposed in [4] to reduce the number of rebroadcasting messages. The authors suggest that it is interesting to privilege the retransmission by nodes that are located at the radio border of the sender. In some sense, their insight of border node retransmission motivates our proposed approach of priority broadcast.

3 Problem Description

3.1 Network Model

We start with the description of the network model of large-scale sensor networks. This is a sensor network consists of a large number of sensors distributed randomly in a monitored disaster region, such as earthquakes, fires, floods, *etc.* Since the sudden configuration changes due to failure, this network is operated in a “zero-configuration” manner and the data collection must be initiated immediately, before the nodes have a chance to assess the current network topology. In addition, the location of the sink is unknown and all data is of equal importance.

Each sensor node has a unique identifier (*ID*) and is capable of sensing an area around itself. And also, each sensor node has a radio interface and can communicate directly with some of the sensors around it. Here, we assume the node number to be N and transmission radius to be R . Specially, it is difficult for the nodes that located far from the sink to deliver their symbols to the sink, since it will take too many hops. Moreover, we assume that each packet can at most contain P symbols, and assume the buffer size to be B , as is depicted in Fig. 1.

Fig. 1. Storing symbols of multi-snapshot in memory ($P = 3$, $B = 5$)

Since the location of the sink is unknown, thus the intermediate node randomly chooses a neighbor and a random packet in its memory and then disseminates this packet to the selected neighbor. In Section 5, we also investigate the technique of priority broadcast, where the intermediate node chooses the packet with the highest priority and then disseminates this packet to all the neighbors.

3.2 Overview of Growth Codes

Growth Codes define the *degree* of a codeword to be the number of packets XOR'd together to form a codeword. Each node initializes its buffer with its own packet (*e.g.*, the sensed information); when a new packet is received, the node stores the incoming packet at a random position in its memory, overwriting a random unit of previously stored information (but not its sensed information).

Though a high codeword degree would increase the probability that a codeword provides innovative information, more codewords and more operations are necessary to decode a received codeword. As is described in [7], codewords in the network should start with degree one and then increase the degree over time. Specifically, for the first $R_1 = \lceil \frac{N-1}{2} \rceil$ packets, codewords of degree 1 are disseminated; after recovering $R_k = \lceil \frac{kN-1}{k+1} \rceil$ packets, codewords of degree $k + 1$ are disseminated.

Let d be the degree of a codeword to be transmitted after r packets have been recovered at the sink, and assume that all codewords have the same probability to be received by the sink, Growth Codes give us:

$$d = \lfloor \frac{N+1}{N-r} \rfloor. \quad (1)$$

4 Maximizing Growth Codes Utility by Coding on The Symbol Level

4.1 Decimal Codeword Degree

Growth Codes assume that each transmission packet contains only one sensed symbol, thus the codeword degree in Growth Codes is always the integer number. However, in many scenarios of multi-snapshot, a sensor node can store multiple symbols in the buffer, thus a transmission packet would contain several sensed symbols. Therefore, when using Growth Codes on the symbol level, to generate the codeword of the decimal degree is indeed possible.

Fig. 2. An example of a codeword with the decimal degree 1.5

Fig. 3. An example that the decimal degree outperforms the integer degree

An example of the decimal degree is depicted in Fig. 2. An intermediate node stores 2 packets (*i.e.*, packet *A* and packet *B*) in the buffer, each packet contains 2 symbols (*i.e.*, a total of 4 symbols: $A_1, A_2; B_1, B_2$). To transmit, this node generates a new packet, which is encoded over one whole packet (A_1, A_2) and another half packet (B_1), to get a new codeword of degree 1.5. Here, the value of 1.5 is calculated by $\frac{1}{2} * 2 + \frac{1}{2} * 1$, since $A_1 + B_1$ has a degree of 2 while A_2 has a degree of 1.

In Growth Codes, after $R_k = \lceil \frac{kN-1}{k+1} \rceil$ packets having been recovered at the sink, a codeword of degree $k+1$ is more likely to be successful decoded than a codeword of degree k . However, when the decimal degree is used, a decimal degree such as $k+0.5$ may outperform both degree of k and $k+1$. For clarity, the example that the decimal degree outperforms the integer degree is given in Fig. 3. It is a 4 node sensor network, each node has sensed 2 symbols and has the same transmission capacity of 2 symbols. At one time, the sink has already recovered 2 packets (see Fig. 3a: A_1, A_2, A_3, A_4), then we compare three codeword degrees: degree 1, degree 2 and degree 1.5.

In the first case (see Fig. 3b), the sink randomly receives a codeword of degree 1. At expectation, 0.5 new packet (1 new symbol) can be recovered. In the second case (see Fig. 3c), the sink randomly receives a codeword of degree 2. At expectation, 0.5 new packet (1 new symbol) can also be recovered. In the third case (see Fig. 3d), the sink may get 2 symbols (*e.g.*, from a codeword of $A_1 + C_1$ and C_2) at the probability of $\frac{1}{3}$, get 1 symbol (*e.g.*, from a codeword of $C_1 + D_1$ and C_2) at the probability of $\frac{1}{2}$ and get no symbol (*e.g.*, from a codeword of

$A_1 + B_1$ and A_2) at the probability of $\frac{1}{6}$. At expectation, the number of newly recovered symbols is: $\frac{1}{3} * 2 + \frac{1}{2} * 1 + \frac{1}{6} * 0 = \frac{7}{6}$, which is larger than 1.

4.2 Search for Appropriate Codeword Degree

Growth Codes select the codeword degree as $d = \lfloor \frac{N+1}{N-r} \rfloor$, but the previous subsection suggests that the codeword of the decimal degree may have a better performance. To get the appropriate decimal codeword degree, we use the approach of interpolation. Notice that, the degree transition points suggested in Growth Codes, which can be thought as the sample set used for interpolation, are: $\{< d, f(d) > | 1 \leq d \leq N, f(d) = N - \frac{N+1}{d}\}$. Then, the problem of searching for appropriate codeword degree is turning into the interpolation problem as follow.

Given a partition $\Delta: \{i = 0, 1, \dots, N-1, x = \{x_i\}, x_i = i+1, y = \{y_i\}, y_i = N - \frac{N+1}{x_i}\}$, how to get the interpolation function $s_\Delta(x, y)$?

Let $s_i = \frac{y_i - y_{i-1}}{x_i - x_{i-1}} = y_i - y_{i-1}$ ($i = 1, \dots, N-1$) and define $m_i = \min(s_i, s_{i+1})$, the method of Hermite spline interpolation [6] tells us:

$$s_\Delta(x, y) = m_{i-1}(x_i - x)^2(x - x_{i-1}) - m_i(x - x_{i-1})^2(x_i - x) \\ + y_{i-1}(x_i - x)^2[2(x - x_{i-1}) + 1] + y_i(x - x_{i-1})^2[2(x_i - x) + 1] \quad (2)$$

The next theorem illustrates the efficiency of Hermite spline interpolation. And the example of this kind of hermite interpolation is illustrated in Fig. 4. For clarity to observe, we only show the part of degree less than 10.

Theorem 1. *The codeword degree distribution calculated by Eqn. 2 is near-optimal.*

Proof. (1) It is obvious that $s_\Delta(x_i) = y_i$, for $i = 0, 1, \dots, N-1$;

(2) Since $s'_\Delta(x_{i-1}) = m_{i-1}$, $s'_\Delta(x_i) = m_i$ and $s'_\Delta(x) > 0 (x_{i-1} < x < x_i)$, thus in each interval $[x_{i-1}, x_i]$, $s_\Delta(x, y)$ is monotonically increasing.

Thus, Eqn. 2 is conform to the “growth” property of Growth Codes, where the codewords degree is gradually increasing as time goes, so that the codeword degree distribution calculated by Eqn. 2 is near-optimal. $\square$

Having known the appropriate degree, then we have the modified coding strategy as follow. When an intermediate node has a chance to disseminate a packet, it first determines an appropriate codeword degree of d and then generates P coded symbols, among which $(d - \lfloor d \rfloor) * P$ coded symbols are encoded over $\lceil d \rceil$ symbols and $(1 - d + \lfloor d \rfloor) * P$ coded symbols are encoded over $\lfloor d \rfloor$ symbols.

5 Maximizing Growth Codes Utility by Priority Broadcast

5.1 Unicast

Since the location of the sink is unknown, and since all data is of equal importance, Growth Codes use a simple algorithm to exchange information: a node

Fig. 4. Searching the decimal degree by interpolation (N=500)

randomly chooses a neighbor and a random codeword in its memory and exchanges information. However, this kind of unicast may be less efficient than broadcast, since the natural property of broadcasting for wireless sensor nodes. Furthermore, the method of unicast is locally uniform and thus is damage for those nodes with long distance from the sink to deliver their packet to the sink. Clearly, it violates the assumption by the design of Growth Codes that all codewords in the network have the same probability to be received by the sink.

The above assumption by Growth Codes only holds if the sink encounters other nodes with uniform probability. However, in less random scenarios, coding performance will decrease, thus to exchange information by unicast is not an efficient way to maximize the utility of Growth Codes. (The simulation results are shown in Section 6.)

5.2 Priority Broadcast

Since the natural property of broadcast in wireless sensor networks, and since broadcast can diffuse a message from a source node to other nodes in the network quickly, broadcast seems to be an intuitive method of more efficient information exchange. To maximize the Growth Codes utility, we use priority broadcast to disseminate packets. It is privilege to deliver the packet with the higher priority.

Firstly, the priority relates to the intersection of the radio areas of two nodes. The packet received from a neighbor that is located at the radio border of an intermediate node should have a high priority, since this kind of border data transmission is beneficial to disseminate sensed data quickly.

It is observed that the distance between two nodes with full duplex communication can be evaluated by comparing their neighbor lists. When two nodes A and B can contact each other, the union of their communication areas $Z_A \cup Z_B$ can be partitioned in three zones (Fig. 5): Z_1 , Z_2 , Z_3 , where Z_1 denotes $Z_A - Z_B$, Z_3 denotes $Z_B - Z_A$ and Z_2 denotes $Z_A \cap Z_B$. We define the ratio μ_1 by: $\mu_1 = \frac{Z_2}{Z_1 + Z_3}$.

Fig. 5. Intersection of the radio areas of node A and node B

Fig. 6. Example of the priority function convexity according to δ parameter

The parameter μ_1 approximately denotes the distance between node A and B , where a larger value of μ_1 denotes a longer distance while a fewer value of μ_1 denotes a shorter distance. When node A wants to know μ_1 , it has to identify its neighbors and the neighbors of B . For that purpose, each node which forwards a broadcast adds the identities of all its neighbors in the message. When node B receives the broadcast message, it compares the list from the incoming message to its own neighbors list. Then it can determine the approximate value of μ_1 . Therefore, when node B receives a packet of sensed data from node A , node B calculates the priority of this packet as follow:

$$p_1 = \left(\frac{\mu_1}{M_1} \right)^{\delta_1}. \quad (3)$$

Here, M_1 denotes the constant which represents the maximal value of μ_1 . This value can be evaluated by the maximal value of the ratio $\frac{Z_2}{Z_1 + Z_3}$ which correspond to the case when the distance between node A and node B is equal to the transmission radius ($M \approx 0.6$). δ_1 denotes a coefficient which is to control the impact of μ_1 (see Fig. 6).

Secondly, the priority relates to the walk length. Here, the *walk length* is referred to as the number of hops for a source packet to take to arrive at certain intermediate node. We can set a counter μ_2 for each source packet and increase the counter by one after each transmission unless it reaches to the sink. Therefore, when a packet is received by an intermediate node, its priority is calculated as follows:

$$p_2 = \left(\frac{\mu_2}{M_2} \right)^{\delta_2}. \quad (4)$$

Here, M_2 denotes the constant which represents the maximal value of μ_2 . Since each node can store at most B packets in the buffer, a data disseminating walk would be interrupted at a random node at the probability of $\frac{1}{B}$. Thus, the value of M_2 can be approximately evaluated by the buffer size B . δ_2 denotes a coefficient which is to control the impact of μ_2 (also see Fig. 6).

Combining the previous equations, each packet in the buffer has the broadcast priority as follows. Theorem 2 shows the efficiency of priority broadcast.

$$p = \left(\frac{\mu_1}{M_1}\right)^{\delta_1} + \left(\frac{\mu_2}{M_2}\right)^{\delta_2}. \quad (5)$$

Theorem 2. *There exists the appropriate δ_1 and δ_2 to promise that the performance of priority broadcast in the scenarios of less random encounters approaches to the ones of random encounters.*

Proof. Fig. 7 illustrates a scenario of less random encounters, where node S is a intermediate node in the network, and where node B has the longer distance to node S than node A . When node S stores both packet A and packet B in its buffer, it should calculate their priorities p_A and p_B by Eqn. 5: $p_A = 1 + \left(\frac{1}{B}\right)^{\delta_2}$, $p_B = \left(\frac{\mu_{1,B}}{M_{1,B}}\right)^{\delta_1} + \left(\frac{2}{B}\right)^{\delta_2}$. Obviously, there exists some value of δ_1 and δ_2 to promise that $p_A = p_B$. Thus, though node B has a long distance from node S than node A , the priority of their packets are equal by appropriate δ_1 and δ_2 . $\square$

Fig. 7. Priority broadcast in the scenario of less random encounters

Having known the priority of each packet in the buffer, then we have the modified coding strategy as follow. When an intermediate node has a chance to disseminate a packet, the packet k is chosen at the probability of $\frac{p_k}{\sum_{i=1}^B p_i}$.

6 Simulation Results

In this section, we look at how efficient our proposed methods can help the sink recover more data in wireless sensor networks. Our simulation is implemented in about 1000 lines of C++ code. We consider a random wireless sensor network where 500 nodes are randomly replaced in a 1×1 square, each sensor node has a same transmission radius R , thus a pair of sensor nodes are connected by a link if they are within a distance R .

6.1 Impact of Coding on the Symbol Level

First, we look at how the codewords with the decimal degree (coding on the symbol level) impacts the data persistence. In our simulation, we assume $P = 10$ and $B = 100$, *i.e.*, each node can store at most 100 symbols (10 packets) in the buffer. The transmission radius R is 0.2.

In the scenarios of multi-snapshot, a transmission packet may contain several sensed symbols. In this case, Growth Codes on the symbol level can promise the codeword degree to gradually increase in the granular level than coding on the packet level. As is depicted in Fig. 8, the sink needs to use 365 codewords to recover 50 percent of symbols by coding on the symbol level but 400 codewords by coding on the packet level. When the sink has received 500 codewords, it can recover 59% of original symbols by coding on the symbol level but only 52.6% of symbols coding on the packet level. Thus, our proposed method persist much more data than the original Growth Codes and the efficiency of coding on the symbol level in the scenarios of multi-snapshot is shown.

Fig. 8. Data persistence in a 500 node random sensor network ($R = 0.2$, $B = 100$)

6.2 Impact of Priority Broadcast

Second, we look at how the information exchange by priority broadcast impacts the performance of Growth Codes in different scenarios, whatever the scenarios of less random encounters or random encounters. We simulate several networks where the node density varies from 10 to 30. Moreover, two kinds of network scale of 500 and 1000 are considered respectively, we assume $\delta_1 = 2$ and $\delta_2 = 0.5$.

In the scenarios of random encounters (*e.g.*, a much dense network where node density is 30), the original Growth Codes perform well by both unicast and priority broadcast. However, in the scenarios of less random encounters, the performance of the original Growth Codes decrease since it is difficult for a part of symbols to reach to the sink. While associating Growth Codes with

(a) Time taken to recover 50% of the data in different networks

(b) Time taken to recover 80% of the data in different networks

Fig. 9. The performance of Growth Codes differs in different networks

priority broadcast, even the symbols that have the long distance from the sink can also reach to the sink. Fig. 9 depicts the time taken to recover certain fraction of sensed data when the nodes use Growth Codes, by both unicast and priority broadcast. The bottommost curve in either Fig. 9a or Fig. 9b plots the theoretical performance of Growth Codes. The results suggest that the performance of the original Growth Codes differs in different networks. If the network scale is large (*e.g.*, $N = 500$ or 1000) but the node density is not large enough (*e.g.*, less than 30), the performance of the original Growth Codes would decrease. However, to associate Growth Codes with priority broadcast can approach to the theoretical performance in all kinds of scenarios. Thus, the efficiency of priority broadcast is shown.

7 Conclusion

In this paper, we generalize the application scenarios of the original Growth Codes to include multi-snapshot and less random encounters, so as to achieve a better performance over a wider range of networks applications, especially for large-scale of wireless sensor networks. The core idea of this paper is: (1) to use the decimal codeword degree instead of the integer codeword degree to deal with the multi-snapshot scenarios; (2) to use the priority broadcast, which is correlated with the node border transmission and the walk length, to approach to the theoretical performance of Growth Codes in the scenarios of less random encounters. In addition, the proposed methods are compared with the original Growth Codes by simulations, and the results validate the efficiency of our proposed methods. We believe that our work would greatly help to the practical application of Growth Codes in kinds of large-scale sensor networks.

Acknowledgement

This work was supported in part by 863 program of China under Grant No. 2009AA01A348, National Key S&T Project under Grant No. 2010ZX03003-003-03, Shanghai Municipal R&D Foundation under Grant No. 09511501200, the Shanghai Rising-Star Program under Grant No. 08QA14009. Xin Wang is the corresponding author.

References

1. Acedanski, S., Deb, S., Medard, M., Koetter, R.: How good is random linear coding based distributed networked storage. In: NetCod (2005)
2. Ahlswede, R., Cai, N., Li, S.Y., Yeung, R.: Network information flow. *IEEE Transactions on Information Theory* 46(4), 1204–1216 (2000)
3. Aly, S.A., Kong, Z., Soljanin, E.: Fountain codes based distributed storage algorithms for large-scale wireless sensor networks. In: IPSN 2008: Proceedings of the 7th international conference on Information processing in sensor networks, pp. 171–182. IEEE Computer Society, Washington (2008)
4. Cartigny, J., Simplot, D.: Border node retransmission based probabilistic broadcast protocols in ad-hoc networks. In: HICSS 2003: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS 2003) - Track 9, p. 303. IEEE Computer Society, Washington (2003)
5. Dimakis, A., Prabhakaran, V., Ramchandran, K.: Decentralized erasure codes for distributed networked storage. *IEEE Transactions on Information Theory* 52(6), 2809–2816 (2006)
6. Ahlberg, J.H., Nilson, E.N., Walsh, J.L.: The theory of splines and their applications. Academic Press, New York (1967)
7. Kamra, A., Misra, V., Feldman, J., Rubenstein, D.: Growth codes: maximizing sensor network data persistence. In: SIGCOMM 2006: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 255–266. ACM, New York (2006)
8. Lin, Y., Liang, B., Li, B.: Data persistence in large-scale sensor networks with decentralized fountain codes. In: 26th IEEE International Conference on Computer Communications, INFOCOM 2007, pp. 1658–1666. IEEE, Los Alamitos (May 2007)
9. Liu, J., Liu, Z., Towsley, D., Xia, C.H.: Maximizing the data utility of a data archiving & querying system through joint coding and scheduling. In: IPSN 2007: Proceedings of the 6th international conference on Information processing in sensor networks, pp. 244–253. ACM, New York (2007)
10. Munaretto, D., Widmer, J., Rossi, M., Zorzi, M.: Resilient coding algorithms for sensor network data persistence. In: Verdone, R. (ed.) EWSN 2008. LNCS, vol. 4913, pp. 156–170. Springer, Heidelberg (2008)