Implementation of Cooperative Information Storage on Distributed Sensor Boards

Anders Grauballe, Mikkel Gade Jensen, Achuthan Paramanathan, Janne Dahl Rasmussen,
Tatiana K. Madsen and Frank Fitzek
Aalborg University, Department of Electronic Systems
Niels Jernes vej 12 A, 9220 Aalborg East, Denmark
Email: (agraubal, mgade, apku04, jannedr, tatiana, ff)@es.aau.dk

***Abstract*— Reliable data retrieval from a sensor network can require a temporal distributed data storage of the measured data among all motes. This is the case when a source of information, or a gateway, is only occasionally present in the network. We consider a network consisting of motes that in order to save energy independently of each other go into offline mode, switching their radios off. We propose a cooperative mechanism for data distribution that increases system reliability, and at the same time keeps the memory consumption for data storage low on each device. A Reed-Solomon coding scheme is applied as a solution where a certain number of unavailable devices can be tolerated without jeopardizing the system reliability. A prototype of a system is implemented to test the protocol behavior under realistic conditions. Prototyping is done on the** `OpenSensor` **platform developed at Aalborg University. To observe the system performance in different settings, additionally a simulator is developed. Performance results indicate that the proposed cooperative data distribution method outperforms the non-cooperative one. The metric used to estimate the system reliability is the probability to retrieve all sensor measurements.**

I. Introduction

Wireless sensor networks are gaining more and more attention and they are expected to be widely used in communication applications, such as intelligent home control systems or safety control in industries. A sensor in these systems could be a small device with the purpose of measuring environmental conditions such as humidity, pressure, temperature etc. and being able to communicate and relay this information to a control unit or gateway. Some applications require certain level of reliability, e.g. in 95% of cases the measured data shall be retrieved. At the same time, sensor devices typically have a limited memory capacity. The limitation on storage capacity of a single node can make it necessary to distribute the data to store it cooperatively on many sensor devices. Data replication can also be applied to increase reliability of data retrieval process. Information storage in wireless sensor networks is the main focus of this paper.

Another important aspect of wireless sensor networks (WSN) consisting of small battery powered devices, is energy consumption. Mechanisms that optimize sensor energy utilization are considered necessary in WSN. One of the most promising approaches is a power-sleeping mode, where devices alternate between active and sleep mode (see e.g. [1], [2]). In [3] it is shown that the combination of both radio off and microcontroller power down mode can significantly increase the network life time. However, saving the energy consumption by letting the devices to sleep can potentially affect the robustness of the system in terms of availability of the measured data. This is another reason why a distributed data storage mechanism is desired in WSN.

The scenario, considered in the paper, is a cluster of sensors, often called motes, which are within communication proximity of each other. The devices are periodically powered off to save energy. The data measurements are occasionally collected by a gateway, or a fusion centre, that is not always present as a part of the system. It means that if at the time of the gateway presence some motes are offline, they are not able to send their data to the gateway (see Fig. 1). Examples that motivate this scenario are a technician coming to collect measurements; an airplane collecting data from a sensor network distributed in a deserted and unreachable area and many more.

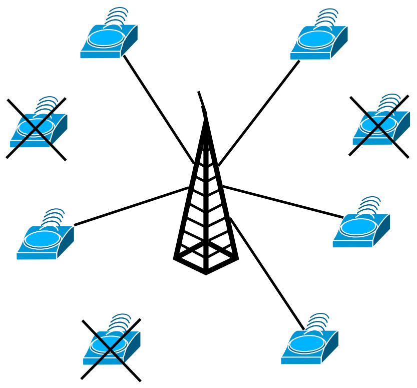

Fig. 1. This figure shows the basic scenario which consists of a gateway and sensors all in range of each other.

If each of the sensors in the cluster stores their own information, thus, there is no cooperation between the devices; the resulting system will be unreliable as the gateway will miss the measurements from inactive sensors. To improve the reliability, we advocate the cooperative approach in this paper. Recently, there has been great interest in cooperative communication [4]. One way to exploit the cooperative diversity in WSN is to create a pool of common resources that can be used to enhance reliability of the transmission link (see e.g. [5]). This paper deals with a problem how cooperation can be utilized in order to achieve the distributed data storage and the subsequent reliable data retrieval. We develop a cooperative method, where motes distribute their data to other motes in a one-hop network such that data from all motes can be received by a gateway even if not all motes are online/in range at a given time. Furthermore, to evaluate the performance of the proposed algorithm, we build a testbed based on the `OpenSensor` platform [6].

The reminder of the paper is organized as follows. In Section II we introduce the proposed distribution method. Section III presents the implementation details, including the description of the `opensensor` platform. We provide the performance evaluation results in Section IV. Finally, we give our conclusions in the last section.

II. Proposed Algorithm

To increase the probability to retrieve a data measurement one can distribute a full copy of the measurement to other motes in the network (or in the cluster - the problem of clustering is not discussed here; it is assumed that the clusters are already formed). However, this approach is not optimal, since it is necessary to consider storage capacities on the devices as well. A suitable method in relation to reliability and memory consumption is the Reed-Solomon (RS) coding scheme which is a technique used to ensure reliable data and to make systems fault-tolerant. Examples of the use of RS are in error detection/correction of CDs, DVDs, RAID storage and DVB systems [7].

In RS l is defined to be the number of data devices and m is the number of redundant checksum devices. The total number of devices in the system is $n = m + l$. The data on each device is divided into words w that is the word size in bits. The RS coding is performed as a linear combination of these words and checksum words on the checksum devices. In RS it is possible to recover the data if up to any m devices in the system are missing. An example could be a system with 4 devices each holding 4 bits of information, so l = 4 and w = 4. If the goal is to recover the devices in the case where any 3 devices fail, then the number of checksum devices must be m = 3. The system then consist of 7 devices and a controller (gateway) which detects how many and which devices will fail.

A. Algorithm for data distribution

Each mote collects the data individually. When the data is collected, it is replicated and distributed to each of the motes in the cluster. The distribution of data from a sensor to other motes in the cluster is done in the following way applying RS coding in the system:

1) Data is split into $l = n - m$ parts
2) m redundant parts are generated using RS on the data parts resulting in a total of n parts
3) The parts are distributed to n motes (including itself), one part for each mote

The steps in the algorithm are illustrated in Fig. 2.

When the gateway arrives to collect data, it is done in the following way:

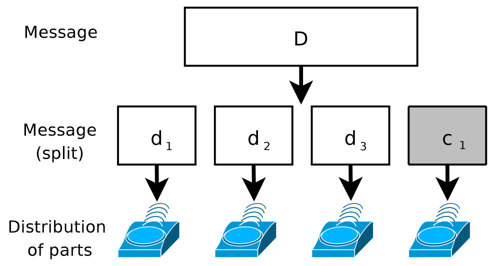

Fig. 2. In this case a message is split into three data parts (d_1 to d_3) and one redundant data part (c_1) is added. Three of the parts are distributed to other motes and one is kept on the mote itself.

1) The gateway broadcast a signal to announce its presence
2) Online motes stay online throughout the gateway session
3) Gateway requests data from each mote. If a timeout occurs on request, the mote is considered offline
4) Gateway receives data from each online mote
5) With the gateway request each mote receives a new session number
6) When the gateway has received data from all online motes, it leaves the cluster
7) The online motes will then measure and distribute new data
8) When offline motes become online, they receive the new session number with new data and deletes old data. Then they will measure and distribute new data

B. Memory consumption

The system consists of n motes in total; the number of checksum motes is m which leaves $n - m$ data motes. Each data mote and checksum mote holds $\frac{1}{n-m}$ of the message. Thus the total memory consumption of one message in the system c_{total} is:

$$\begin{aligned} c_{total} &= \frac{n-m}{n-m} + \frac{m}{n-m} \\ &= \frac{n}{n-m} \end{aligned} \tag{1}$$

This ratio is also the total memory consumption in each mote when holding messages for all other motes (including own message).

Equation 1 can be used to decide the number of checksum motes, if there is a requirement to the maximum memory consumption and the total number of motes is known.

In the proposed solution any arbitrary mote can fail as long as the total number of offline motes does not exceed the number of checksum motes. This property makes RS an ideal coding scheme for the distribution model, because it can provide a good flexible trade-off between reliability and memory consumption and tolerate random failures in the system.

III. Prototype

The aim of this prototype is to design and implement a system model which gives an indication of how the proposed protocol performs in a real life scenario. For this purpose we are using OpenSensor v2.0 platforms developed in Aalborg University [6]. In the following we describe the implementation platform and its properties. Initially, the platform has had a very primitive MAC protocol. In order to achieve higher transmission reliability in wireless environment, the MAC protocol with carrier sensing capabilities has been implemented. Furthermore, we present the implementation procedures for data distribution protocol and describe the testbed.

A. Sensor platform

Each OpenSensor mote contains a microprocessor, communication module, sensor module and power supply. All of this is contained in a box typically about the same size as a small mobile phone.

The OpenSensor v2.0 consists of two different boards stacked on each other which give the flexibility of adding more features in a later design. This mote is the one being used for the implementation and it has the features described in the following.

Main board

- Li-Poly battery interface which can be used for power supply to the mote
- Mini USB interface which can be used for both serial RS-232 interface to the mote and external power supply. It is also possible to charge the Li-Poly battery by changing the jumper settings on the board.
- dsPIC33FJ256GP710 (Microchip) microprocessor for controlling the mote
- 22.1 MHz oscillator as external clock source for the dsPIC
- ICD 2 debugger/programmer interface

Wireless board

- Bluetooth module (Amber Wireless AMB2300) for wireless communication with mobile phones or PCs, connected with serial RS-232 connection to the microprocessor
- nRF905 (Nordic Semiconductor) transceiver for communicating via the ISM band
- Loop antenna for the RF transceiver

A third version (OpenSensor v3.0) has recently been finalized with a few changes to the design to make it easier to assemble without the use of special tools. The design changes include:

- Only one board (no "stacking" needed)
- Cheaper microprocessor (Microchip dsPIC30F3013)
- USB interface replaced by regular RS-232 interface
- Removable antenna
- Programmable LED
- Easy connecter for external I/O equipment

The OpenSensor v3 platform can be seen in Fig. 3.

B. MAC Protocol

Impairments of a wireless medium and its broadcast nature require special MAC protocols that support collision avoidance. Even though the focus of our work is on the data distribution protocols, a proper functioning MAC protocol is

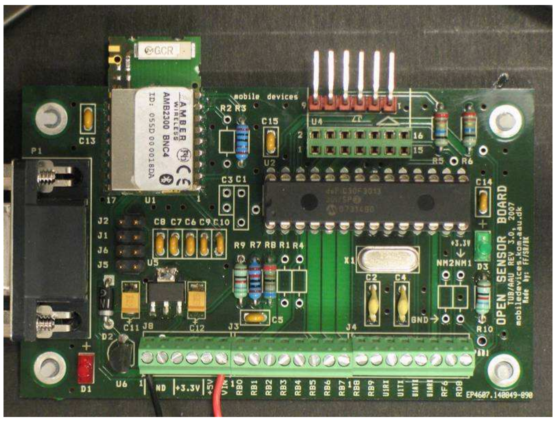

Fig. 3. The new OpenSensor v3 hardware platform

needed for data exchange among the motes and between a mote and a gateway. As we will see during the testing, the behavior of a MAC protocol can impact the behavior of higher layer protocols, that is, data distribution in our case.

The developed MAC protocol features carrier sensing (1-persistent CSMA), a dynamic medium reservation scheme (handshaking mechanism utilizing RTS and CTS control message exchange) known from IEEE 802.11, acknowledgment (ACK) and a one bit sequence number to prevent packet duplicates. The total packet size of the platform is 32 bytes, 3 bytes is used for MAC header which leaves 29 bytes for payload and higher layer protocols. The packet including MAC header bits can be seen in Fig. 4. Notice the two reserved bits for adding extra features like broadcast (i.e. no ACK reply from receivers) which can be useful for future development. The field Packet Type of 4 bits is also not fully used as only 4 types are currently supported: RTS, CTS, Payload and ACK.

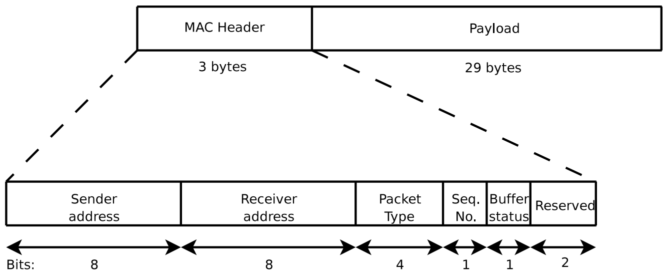

Fig. 4. The figure shows a packet with the MAC header.

C. Design

Each of the distinct prototype motes has a routine which includes executing several tasks during a given period of time and those tasks are defined as follows:

1) **Measure Data**
   The objective of this task is to measure (generate) a data packet with a maximum size of 29 bytes (maximum allowable due to platform specifications).
2) **Encode Data**
   This task encodes the measured data by adding redundant data and split the packet into three parts for distribution to other motes.

3) **Distribute data**
The encoded data is distributed to two other motes by accessing the MAC-protocol services routine, and the last data packet is stored in its own memory.
4) **Online/offline**
The online period is set to 5 seconds. If the distribution is completed within this period, the mote will enter an idle state where it will listen to the radio channel for incoming messages. At the end of each online period a mote will make a decision whether it stays online or offline during the next 5 sec period. We refer to the task of making the decision as Probability Decision.
5) **Probability Decision**
Decision to stay online or to go into offline mode is made by each sensor individually and independently from other sensors. With the probability p_{ON} a mote is in online mode. Probabilities p_{ON} are the same for all motes and they do not change with time.
6) **Listen**
During the idling, the mote will listen for incoming messages e.g. message from gateway or incoming data packet from other motes. In case of a message from a mote, the incoming packet will be stored in the memory.
7) **Gateway presence**
When a mote receives a presence message from a gateway it will enter a "freeze" state, which means that the mote will do nothing but listen to gateway. When it receives a request message from the gateway it will begin to transmit the collected measurements.

A state diagram for one mote in the system can be seen in Fig. 5.

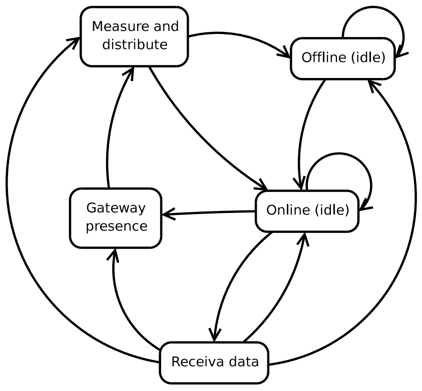

Fig. 5. State diagram for one mote

In order to schedule those above mentioned tasks a kernel is implemented on the `OpenSensor` v2.0 platform.

The gateway is designed such that it includes two tasks:

1) **Announcement**
The gateway announces its presence every 15 seconds. After the gateway presence is broadcasted to the network, the online motes are in "freeze" state.
2) **Information gathering**
After the presence has been broadcasted by the gateway, it will begin to send a request messages to the motes and gather the data.

D. *Testbed*

The testbed consists of four devices, where one is used as a gateway and three as motes. It is shown on Fig. 6. Different light signals are used to visualize the mote states (receiving, sending, sleeping, distributing).

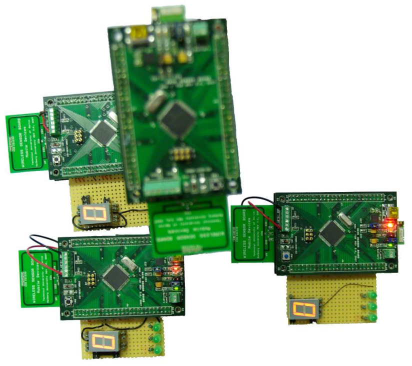

Fig. 6. This figure shows the system prototype which consists of three sensors and one gateway.

IV. Results

A. *Prototype test result*

One sample in the prototype test is defined as the result of one data gathering procedure performed by the gateway. The outcome of the test is an answer of whether the reconstruction of the message was possible or not. Three different test cases with distinct online probabilities = {0.5, 0.8, 0.9} are tested. In each test case 200 samples are collected. Because only 4 devices are available the test case is performed for the algorithm based on the RS(3,1) coding scheme.

The conducted results from these cases showed e.g. that for an online probability of 0.5, the gateway was able to reconstruct 85 times out of 200 runs. The result from this test can be seen in Table I.

Online probability	**Reliability [%]**
0.50	42.5
0.80	60.5
0.90	79.5

TABLE I
The results from the prototype tests showing the reliability at three different online probabilities

B. *Simulation and analytical results*

Due to the limitation in the number of motes in the prototype, it was possible to test the performance of the proposed protocol only with RS(3,1) scheme. In order to understand how the choice of the coding scheme influences the data retrieval reliability, we have investigated the system performance by means of a simulation and using analytical approach.

We have implemented a simulation of the system in Java both to investigate the performance of the system with different number of devices and to illustrate the different states that each mote would be in at a given time. The parameters that can be varied are:

- Total number of motes

- Number of checksum motes
- Online probability
- Period time for the mote

To ensure that the motes are independent regarding state shift, each mote runs in a separate thread. Furthermore, an ideal MAC protocol is assumed, that is each sent packet is received without any errors.

Each of the following cases has been tested with online probability = {0.5, 0.7, 0.8, 0.9} :

- RS(3,1)
- RS(3,2)
- RS(21,7)
- RS(21,14)
- RS(48,16)
- RS(48,32)

In each case 1000 samples are collected. The results can be seen in Table II.

RS / p_{on}	(3,1)	(3,2)	(21,7)	(21,14)	(48,16)	(48,32)
0.5	0.494	0.859	0.107	0.972	0.013	0.993
0.7	0.774	0.97	0.775	1	0.738	1
0.8	0.903	0.993	0.948	1	0.992	1
0.9	0.969	1	0.999	1	1	1

TABLE II

The results from the simulation tests showing the reliability at four different online probabilities (p_{on}) for six RS schemes.

C. *Comparison of prototype and simulation results*

Comparing Table I and Table II, we observe that the results from the prototype test show deviation from the simulation results. The measured probability to retrieve a message is lower compared with the simulated case. This deviation can be explained by the fact that ideal MAC protocol is used for the simulation model. Even though in the implemented MAC protocol collision avoidance mechanisms are implemented, the packets can still be lost, thus resulting in the degradation of the data dissemination and data retrieval protocol. We can conclude that the implemented MAC protocol still needs fine tuning.

D. *Comparison of cooperative and non cooperative approaches*

Fig. 7 shows the probability that the gateway is able to reconstruct a message if cooperative data distribution algorithm is applied (solid lines) and if there is no cooperation among the motes in the cluster (dashed lines). The curves are constructed for different parameters of the RS coding scheme. These results confirm our intuition that using the cooperative method using Reed-Solomon is better than the non-cooperative case. For example in Reed-Solomon, reliability at 50 % can be guaranteed with only 0.50 online probability where the non-cooperative case needs an online probability at approximately 0.80 to ensure the same reliability. An online probability in both methods at 0.70 gives a reliability in Reed-Solomon at 78.5 % and in the non-cooperative case at 35 %.

Furthermore it can be seen that distributing the message among larger number of nodes (corresponding to the higher parameters of the coding scheme) helps to achieve high reliability even for reduced p_{ON}.

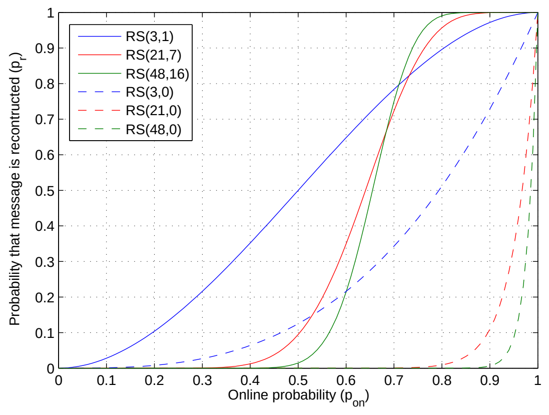

Fig. 7. Each solid line represents a cooperative approach using Reed-Solomon coding RS(x,y) with a ratio of x:y. The corresponding dashed lines show non-cooperative approach.

V. Conclusion

The objective of this paper is to propose a reliable solution that enables distribution of measured sensor data to a gateway in a partially wireless connected sensor network. The approach when motes cooperate to store the collected data has a clear advantage compared with a non-cooperative approach, when each mote stores its own measurements. Distributed storage achieves high probability to retrieve all stored data. The Reed-Solomon coding scheme has been selected as it enables the gateway to recover data with a high reliability and at the same time it is efficient in terms of memory requirements for data storage.

The proposed protocol has been implemented and tested on `OpenSensor` platform. The initial testbed consists of 3 motes and 1 gateway. At the time of writing the paper, we are extending the testbed with a larger number of motes which enables us to test different settings of the protocol varying the parameters of Reed-Solomon coding scheme.

References

[1] W. Awada, M. Cardei. Energy-Efficient Data Gathering in Heterogeneous Wireless Sensor Networks. IEEE Intl. Conf. on Wireless and Mobile Computing, Networking and Communications (WiMob'06), Jun. 2006.

[2] B. Yin, H. Shi, Y. Shang. Analysis of Energy Consumption in Clustered Wireless Sensor Networks. 2nd International Symposium on Wireless Pervasive Computing, February 2007.

[3] A. Minhas, T. Trathnigg, C. Steger, R. Weiss. Energy Saving in Pervasive Sensor Networks. 2nd IET International Conference on Intelligent Environments, July 2006.

[4] F.H.P. Fitzek and M. Katz. Cooperation in Wireless Networks: Principles and Applications – Real Egoistic Behavior is to Cooperate!. 2006. Springer.

[5] T. Quek, D. Dardari, M. Win. Energy Efficiency of Dense Wireless Sensor Neyworks: To Cooperate or Not to Cooperate. IEEE Jour. on Selected Areas in Communications. Vol.25, No. 2, February 2007.

[6] *Mobile Devices - OpenSensor*. `http://mobiledevices.kom.aau.dk/opensensor/`

[7] Robert H. Morelos-Zaragoza. The Art of Error Correcting Coding. 2006. John Wiley and Sons Inc.