Distributed Data Delivery in Partially Connected Networks

Anders Grauballe, Mikkel Gade Jensen, Achuthan Paramanathan and Janne Dahl Rasmussen
Supervisor: Tatiana Kozlova Madsen and Co-supervisor: Frank Fitzek
Aalborg University, Department of Electronic Systems 2007
Email: (agraubal,mgade,apku04,jannedr)@es.aau.dk

***Abstract*— Wireless sensor networks are expected to be widely used in communication and control systems in the future. Some applications introduce only partially connected sensor networks leading to lower reliability or availability of the measured data. The objective of this project is to develop and analyze a data distribution method which can increase the system reliability and keep the memory consumption 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 probabilistic system model is derived to describe the distribution and reconstruction of a message from a sensor to the gateway. This model is verified and visualized by means of a simulation implemented in Java. A prototype of a system containing two data devices and one redundant device is implemented to test the model in a real life scenario. Performance evaluation shows that the probability to receive all sensor measurements by using the proposed cooperative data distribution method is higher compared with the non-cooperative case.

I. Introduction

Recently, wireless sensor networks have been introduced and 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 a certain level of reliability and/or a maximum of memory consumption which is important to consider in this project.

The basic scenario is a cluster of sensors, often called motes, which are partially wireless connected with a gateway, which means that the system reliability in the network is not certain. In such networks of small battery powered devices, energy consumption is a critical topic and devices are often periodically powered off to save energy. This means that each mote will have periods where it is online as well as offline. The gateway is connected with every online mote, but is only present when data should be collected. The result is that the gateway in this scenario will not always be able to receive each measurement from the sensors, see Fig. 1.

In this non-cooperative case, each of the sensors in the cluster stores their own information, which produces a very unreliable system as the gateway will miss the measurements from inactive sensors.

The objective of this project is to develop a cooperative method, where motes distribute their data to other motes in a one-hop network, see Fig. 2, 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. The parameters for the online probability of each sensor and data storage capacity are to be considered in the method.

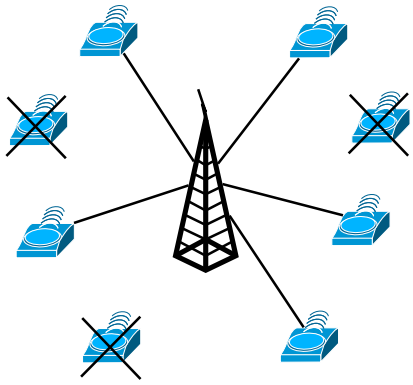

Fig. 1. This figure shows the basic scenario which consists of a gateway and sensors all in range of each other. Sensors are randomly unavailable to the gateway, which can result in failures.

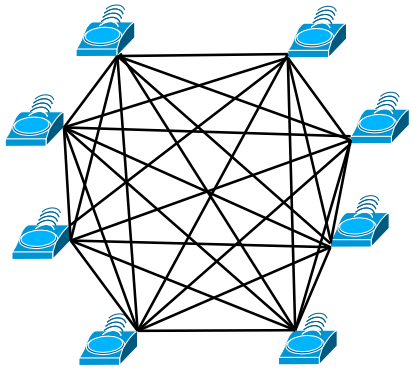

Fig. 2. All sensors are in range of each other and are able to distribute data in order to increase the reliability.

II. Proposed solution

A solution to the reliability problem could be to distribute a full copy of the measurement to other motes in the cluster,

but because it is necessary to consider storage capacities on the devices it is not optimal to distribute the entire message to all other motes in the cluster. 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 [5].

A. Reed-Solomon

In the following subsection a description of how RS can be used in a RAID like system is given [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 devices 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.

The following shows an example on how to recover after failure:

In a system there are 4 devices each holding 4 bits of information, so l = 4 and w = 4. The goal of this example is to recover the devices in the case where any 3 devices fail, so 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.

The mathematical operations multiplication and division is over a Galois Field that is defined to be $GF(2^w)$, addition and subtraction are performed by XOR operations and are denoted by $\oplus$. The operations over Galois Field is in this example performed by a table lookup [7].

The information on the devices are the following:

$$d_1 = 0101 = 5$$

$$d_2 = 1000 = 8$$

$$d_3 = 1101 = 13$$

$$d_4 = 1011 = 11$$

To compute the checksum for the checksum devices a function F is applied to the data devices:

$$c_i = F_i\left(d_1, d_2, ..., d_l\right) = \sum_{j=1}^{l} d_j f_{i,j}$$

F is a $m \times l$ Vandermonde matrix which in this case is calculated like this:

$$F = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & l \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 2^{m-1} & 3^{m-1} & \cdots & l^{m-1} \end{bmatrix} \begin{bmatrix} 1^0 & 2^0 & 3^0 & 4^0 \\ 1^1 & 2^1 & 3^1 & 4^1 \\ 1^2 & 2^2 & 3^2 & 4^2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 5 & 3 \end{bmatrix}$$

When the Vandermonde matrix is found the checksum can be calculated. Multiplications are performed over $GF(2^4)$ and the additions are done by XOR operations.

$$\begin{aligned} c_1 &= (1)(5) \oplus (1)(8) \oplus (1)(13) \oplus (1)(11) \\ &= 5 \oplus 8 \oplus 13 \oplus 11 \\ &= 0101 \oplus 1000 \oplus 1101 \oplus 1011 = 1011 = 11 \end{aligned}$$

$$\begin{aligned} c_2 &= (1)(5) \oplus (2)(8) \oplus (3)(13) \oplus (4)(11) \\ &= 5 \oplus 3 \oplus 4 \oplus 10 \\ &= 0101 \oplus 0011 \oplus 0100 \oplus 1010 = 1000 = 8 \end{aligned}$$

$$\begin{aligned} c_3 &= (1)(5) \oplus (4)(8) \oplus (5)(13) \oplus (3)(11) \\ &= 5 \oplus 6 \oplus 12 \oplus 14 \\ &= 0101 \oplus 0110 \oplus 1100 \oplus 1110 = 0001 = 1 \end{aligned}$$

To recover from failures a matrix A and vector E must be defined. A is a matrix with the identity I matrix in top and the Vandermonde matrix F in the bottom. E is the information on the data and checksum devices.

$$A = \begin{bmatrix} I \\ F \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 5 & 3 \end{bmatrix} \quad E = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \\ 13 \\ 11 \\ 11 \\ 8 \\ 1 \end{bmatrix}$$

In A' and E' the rows and elements of the failing devices are removed (in this case d_2, d_3 and d_4 are lost).

$$A'D = E' \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 5 & 3 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} 5 \\ 11 \\ 8 \\ 1 \end{bmatrix}$$

The system must be solved for D by performing Gaussian elimination or an equivalent operation to get the inverse of A', e.g. by performing the MATLAB operation $inv(gf(A_{prime}, 4))$.

$$D = (A')^{-1}E' \Rightarrow D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & 2 & 6 & 7 \\ 4 & 5 & 7 & 6 \\ 6 & 6 & 1 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 11 \\ 8 \\ 1 \end{bmatrix}$$

To recover the information the following operations are performed:

$$\begin{aligned} d_2 &= (3)(5) \oplus (2)(11) \oplus (6)(8) \oplus (7)(1) \\ &= 15 \oplus 5 \oplus 5 \oplus 7 = 8 \end{aligned}$$

$$\begin{aligned} d_3 &= (4)(5) \oplus (5)(11) \oplus (7)(8) \oplus (6)(1) \\ &= 7 \oplus 1 \oplus 13 \oplus 6 = 13 \end{aligned}$$

$$\begin{aligned} d_4 &= (6)(5) \oplus (6)(11) \oplus (1)(8) \oplus (1)(1) \\ &= 13 \oplus 15 \oplus 8 \oplus 1 = 11 \end{aligned}$$

The information in the system is then recovered.

B. Proposed algorithm

The distribution of data from a sensor to other motes in the cluster is done in the following way to apply 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. 3.

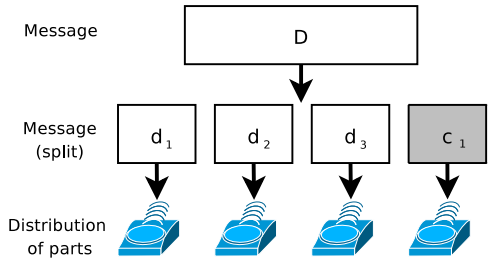

Fig. 3. In this case a message is split into three data parts (d_1 to d_3) and one redundant data part (c_1) is added. Three of the parts are distributed to to other motes and one is kept on the mote itself.

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

1) The gateway broadcast a signal to announce its presence
2) Online motes stays 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

C. 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 RS coding scheme 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. Performance evaluation

In order to evaluate the proposed solution a mathematical model of the system is presented in the following section. First a scenario will be presented in detail, describing the parameters considered by the model. The mathematical model will then be evaluated through a simulation and a prototype test. The performance of RS encoding/decoding algorithm will not be considered in this project, but performance regarding reliability of the system is the main focus.

A. System scenario

This scenario is given to provide an example of the system and to create assumptions about system parameters prior to the modeling.

The scenario will describe these different parameters in the system:

- Time of measurements
- Online/offline pattern
- Arrival time of the gateway

Common assumptions of parameters for the scenarios are:

- Propagation delay = 0
- The online/offline period is fixed
- The probability for a mote being online is p_{on} and offline is $p_{off} = 1 - p_{on}$ (Bernoulli random variable). This applies to all motes independently.
- The system consists of n motes
- The system can tolerate m failures ($m < n$)
- No motes will fail (uncontrolled breakdown) at any time

The motes are all turned on and each will decide to enter online mode with probability p_{on} or offline mode with probability $1-p_{on}$. If a mote is online it will perform a measurement and afterwards encode and distribute this measurement as n packets to the other motes. If some motes are offline the distributing mote will extend its online time until the offline motes becomes online and the packets can be distributed.

After this procedure the mote will again decide whether the next period should be online or offline. This will happen on all motes, so after some time which can be denoted as measurement and distribution time $M + D$, all motes will contain n packets. When distribution is done, the motes will enter the idle phase I, shifting between online and offline mode with probability p_{on} to enter online mode.

The gateway arrives within a time interval G after the $M + D + I$ phase has ended, triggering the online motes to send their

data so the gateway is able to regenerate the measurements from the received packets. When the gateway has left, the $M+D$ phase will start again and the motes which were offline when the gateway was present will delete their packets and start measuring again. This scenario can be seen on a time line in Fig. 4.

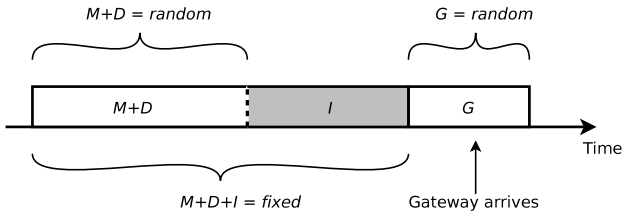

Fig. 4. The time line shows the different phases in one cycle in the system. $M+D$ is a random period and I is the remaining time until the gateway phase G

From this scenario it is seen that the gateway is able to recover measurements if they are correctly distributed within $M+D$, and if enough motes are online at a certain time after the $M+D$ phase. The aim of the modeling is then to find a lower bound of the probability p_{on} which will save battery time and still be able to meet the reliability requirement p_R for reconstruction of a message.

B. Probabilistic model

In this section a mathematical probabilistic system model of the sensor network is derived. The aim of this model is to describe the probability p_R that a gateway successfully will reconstruct the distributed message from the network. In order to do so, some motes have to be online at the same time when the gateway request for data.

1) Data reconstruction probability: By using the RS coding scheme, the system can tolerate loosing as many data parts as there are checksum parts, that is, the probability of a successful reconstruction of the message given a correct distribution.

$$p_r = \Pr(\text{Number of offline motes} \leq m) \Leftrightarrow \Pr(\text{Number of online motes} \geq n-m) \quad (2)$$

As the number of online motes is a series of n independent trials with a probability of success p_{on} and a probability of failure $1-p_{on}$, it is a binomial random variable. [8]

Calculation of p_r given a correct distribution, by binomial distribution

$$\begin{aligned} p_r &= \Pr(\text{Message successfully reconstructed}) \\ &= \Pr(\text{Number of online motes} \geq n-m) \\ &= \sum_{k=n-m}^{n} \binom{n}{k} p_{on}^k (1-p_{on})^{n-k} \end{aligned} \quad (3)$$

The probability of the message being successfully passed to gateway p_r is based on all k online motes with individual online probability of p_{on}.

The following shows an example of a calculation of the probability p_{on} using the binomial distribution. To do this it is assumed that:

- The Reed-Solomon RS(12,4) coding scheme is used (total number of motes is set to n = 12 and m = 4)
- The message must be reconstructed with a probability $p_r = 0.80$.

In the example p_{on} is unknown and X denotes the number of online motes

$$P\{X \geq 8\} = \sum_{k=8}^{12} \binom{12}{k} p_{on}^k (1-p_{on})^{12-k} = 0.8 \quad (4)$$

$$\Leftrightarrow \quad p_{on} = \begin{cases} -0.38 \\ 0.73 \end{cases} \quad (5)$$

According to Equation 5, each mote must be online with a probability of at least 0.73 as a probability is a non-negative number between 0 and 1.

Fig. 5 shows the value of p_{on} as a function of p_r. It can be seen that each mote has to be online with a probability of 0.73 at 80% reliability. The figure also shows the non-cooperative case with 12 motes. In this case motes must be online with probability 0.98 to make the system 80% reliable. From Equation 3 using $m = 0$ the non-cooperative formula is:

$$p_r = p_{on}^n \quad (6)$$

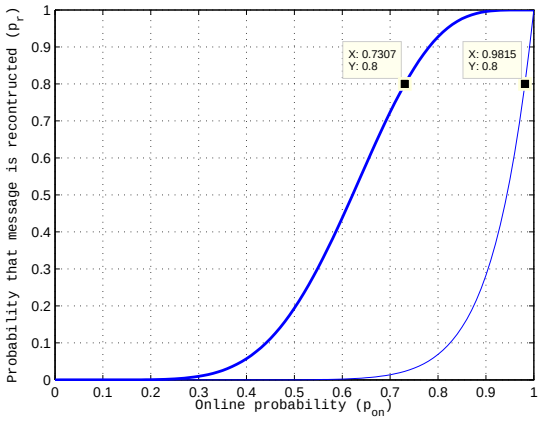

Fig. 5. The graph shows the relationship between the system reliability p_r and the online probability of the individual mote p_{on}. The bold line is RS(12,4) and the thin line is RS(12,0) i.e. the non-cooperative case.

2) Data distribution probability: Probability p_d that one message is distributed to all motes within the $M+D$ phase under the assumption that only one message is present in the system. I.e. a sensor is distributing a message to n motes which are in idle state. If the distribution is not complete within the $M+D$ phase, the data recovery is not possible.

Fig. 6 shows the decision periods of a mote in idle state. This case shows the worst case of a continuously offline mote until $(N-1)T$ which means that the message can still be recovered in the last period where the mote is online.

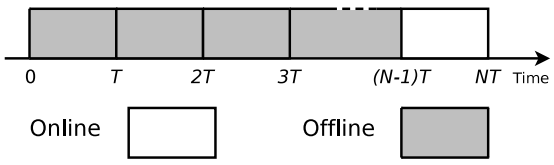

Fig. 6. Worst case of a mote being continuously offline until time $(N-1)T$ and still receive a message before time NT

The probability of successful distribution is also calculated based on the binomial distribution where each trial is a mote being offline all the time within $N \cdot T$ or not. At each period T in time, the mote will make an independent decision with probability p. Thus the probability of the same state (on/off) in N periods is p^N. From this p_d is given as:

$$\begin{aligned} p_d &= \Pr(\text{Message successfully distributed in time } N \cdot T) \\ &= \Pr(\text{All motes online somewhere within time } N \cdot T) \\ &= 1 - \Pr(\text{One or more motes still offline at time } N \cdot T) \\ &= 1 - \Pr(Y \geq 1 \text{ mote still offline at } N \cdot T) \\ &= 1 - \sum_{i=1}^{n} \Pr(Y = i) \\ &= 1 - \sum_{i=1}^{n} \binom{n}{i} (p_{off}^N)^i (1 - p_{off}^N)^{n-i} \\ &= 1 - \sum_{i=1}^{n} \binom{n}{i} (1 - p_{on})^{N \cdot i} (1 - (1 - p_{on})^N)^{n-i} \end{aligned} \quad (7)$$

In Fig. 7 the distribution probability p_d is shown as a function of the online probability p_{on} with 10 different N.

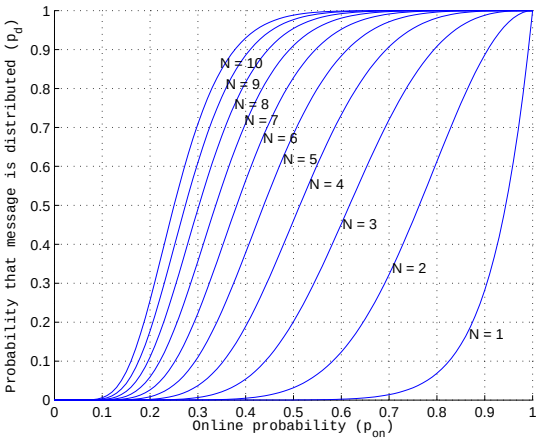

Fig. 7. The probability that the message is distributed within $N \cdot T$ as a function of the online probability p_{on} . The graphs are showing the relationship for different N. Still RS(12,4) is used

3) Total system reliability: From Equation 7 and 3 the total system reliability is:

$$\begin{aligned} p_R &= p_d \cdot p_r \\ &= \{1 - \sum_{i=1}^{n} \binom{n}{i} (1 - p_{on})^{N \cdot i} (1 - (1 - p_{on})^N)^{n-i}\} \\ &\quad \ldots \cdot \sum_{k=n-m}^{n} \binom{n}{k} p_{on}^k (1 - p_{on})^{n-k} \end{aligned} \quad (8)$$

In Equation 8 it is assumed that the events are independent and that the message can not be reconstructed before distribution is complete. Fig. 8 shows the graphs of Equation 8. From the graph it can be seen that the relationship is approximately the same as p_r for large N.

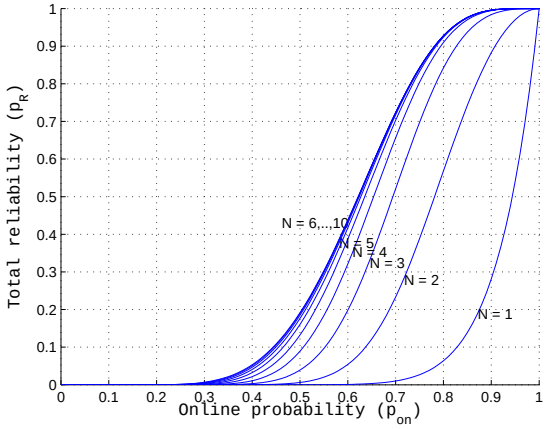

Fig. 8. The total reliability p_R as a function of the online probability p_{on}. The graphs are showing the relationship for different N

The model has some limitations and aspects from the described scenario which are not considered. The model is based on a single mote distributing data to other motes which are in idle state (not distributing). According to the scenario, all motes will try to distribute data in the online period following a gateway presence. This behavior will results in an extended distribution phase. This leads to lower probability for distributing all data parts within a given time.

In the case that not all data parts are distributed before a gateway presence, the message will be lost (failure). It is still possible for the gateway to recover the message in case of partial distribution, provided that the number of missing data parts plus the number of offline motes does not exceed m. This would lead to a higher reliability than the model proposes.

4) Alternative behavior in case of failure: The system reliability can be increased by making a small change of the scenario e.i. if the distribution is not complete when gateway arrives, the distributing motes can just send the whole message to the gateway directly. This yields the following reliability:

$$
\begin{aligned}
p_R &= \Pr(\text{Message successfully reconstructed}) \\
&= \Pr(\text{Distr. successful}) \cdot \Pr(n-m \text{ motes online}) \\
&\quad + \Pr(\text{Distr. unsuccessful}) \cdot 1 \\
&= p_d p_r + (1-p_d) \qquad (9)
\end{aligned}
$$

Fig. 9 shows the graph of the extended scenario of Equation 9 compared with the basic scenario of Equation 8. It can be seen that the reliability is increased for smaller p_{on}, but not for higher p_{on}. It seems that the system reliability goes to 1 as p_{on} goes to 0, but this also means that few motes will be online when the gateway arrives and thus few measurements will be available. Of course if $p_{on} = 0$ no measurements will be available. Using the extended scenario for higher p_{on} does not give any benefits and for lower p_{on} the distribution does not have any influence.

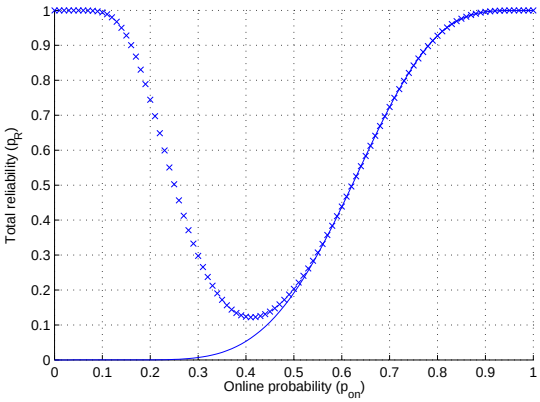

Fig. 9. The total reliability p_R as a function of the online probability p_{on}. The graphs are showing the relationship for $N = 5$. The solid line is Equation 8 and the x marks is the extended scenario of equation 9. RS(12,4)

The derived probabilistic system model describes the reliability of an arbitrary combination of RS coding based on a given online probability. The system model of Equation 8 which reduces to Equation 3 for large N, will now be validated through a system simulation.

C. Simulation

It has been chosen to implement a simulation of the system in Java both to verify the probabilistic model and to illustrate the different states that each mote would be in at a given time. To verify the probabilistic model several parameters must be varied. The parameters can be seen in the list below:

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

To illustrate the different states the motes can be in, first the different state must be defined: In Fig. 10 a state diagram can be seen. From this figure we get the following different states each mote will be in:

1) **Measure and distribute**
   In this state the mote will distribute its data to all other motes. It can receive data at the same time. The mote will not change state until it has distributed to all other motes.
2) **Receive data**
   The mote can receive data when it is online, but also when it is distribution to other motes. Therefore this state is not illustrated in the simulation
3) **Gateway presence**
   In this state the mote sends its data to the gateway.
4) **Online**
   In this state the mote is online. This means that it can be contacted either by another motes because it should receive data, or by the gateway because it should send its data.
5) **Offline**
   The mote can not be contacted when it is offline.

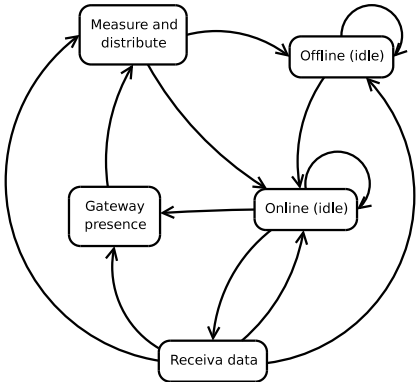

Fig. 10. State diagram for one mote

In Fig. 11 a screenshot of the simulation can be seen. Offline (black dots) and online (green dots) states are shown in this figure. If the gateway arrives the dots will be blue and if a mote is distributing it will be red.

To ensure a correct and realistic simulation it is necessary that the motes are independent regarding state shift. This means that each mote runs in a separate thread, which means the simulation can run with up to approximately 50 motes depending on computation power. Also the gateway must be independent of the motes thus it is also a thread. Each mote must be able to distribute to all other motes and stay online until the distribution is complete. This means that when a mote is asked to receive it must also be able to inform the distributing mote about the present online/offline state. Furthermore each mote must be able to decide upon the online/offline state independently from the previous state according to the online probability. The gateway must be able to request data from each mote and calculate if enough motes are online for the message to be reconstructed.

D. Prototype

The aim of this prototype is to design and implement a system model which gives an indication of how the system performs according to the objective of this project in a real life scenario. For this purpose four openSensor v2.0 platforms are provided by Aalborg University [1], where one is used

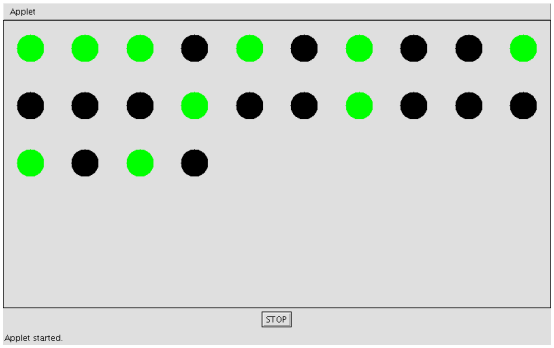

Fig. 11. Screen shot showing the visual part of the simulation. Each dot represents a mote, where green indicates online and black offline. In this case the number of motes is 24

as a gateway and three as motes. A MAC-protocol designed by a previous project group, is used to ensure a reliable data communication between these devices. It features carrier sensing (CSMA), a dynamic medium reservation scheme (RTS and CTS) known from IEEE 802.11, acknowledgment (ACK) and a one bit sequence number to prevent packet duplicates. [2]

The prototype is developed according to the scenario and state diagram described earlier in this paper. Each of those 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 generate a unique data packet of a maximum size of 29 bytes [2].
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. Else it will run the probability decision task.
5) **Probability Decision**
   In this task the above mentioned online / offline probability is handled. That is, for every 5 seconds a mote will undertake a probability control. This is done by generating a random number between 0 and 10. If the random number exceeds a predefined threshold of e.g. 8, the mote will enter the offline state otherwise it will stay online/idle.
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 its received data packets.

In order to schedule those above mentioned tasks a kernel [6] is implemented on the openSensor v2.0 platform for more information see paper worksheets. The prototype with three sensors and one gateway can be seen in Fig. 12.

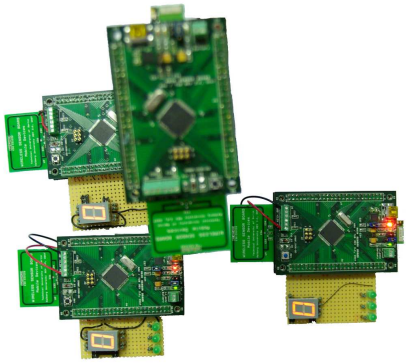

Fig. 12. This figure shows the system prototype which consists of three sensors and one gateway.

The gateway is designed such that it includes two tasks:

1) **Time table**
   The timetable consists of a period of 15 seconds, and at end of each period the gateway will broadcast its presence to the network, and thereby freezing the online motes.
2) **Send request**
   After the presence has been broadcasted by the gateway, it will begin to send a request messages to the motes.

IV. Results

The simulation and prototype have been tested. The simulation is used as an optimal implementation of the model where it is possible to have many samples where the MAC protocol is considered to be error proof. These tests can therefore be used to verify the probabilistic model. The tests from the prototype are used to decide how well the model reflects a real life scenario. One sample in these tests is defined as the result from the gateway in one cycle. In Fig. 4 one cycle is shown. The result from the gateway is an answer of whether the reconstruction of the message was possible or not.

It is assumed that each mote in the network successfully have distributed their messages before the gateway arrives.

A. Simulation

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 conducted results can be seen in Table I.

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 I

The results from the simulation tests showing the reliability at four different online probabilities for six RS schemes.

B. Prototype

Three different test cases with distinct online probabilities: 0.5, 0.8 and 0.9 are tested. In each test case 200 samples are collected. Because only 4 devices are available the test case will be the RS(3,1).

The conducted results from these cases showed that for an online probability of 0.5, the gateway was able to reconstruct 85 times out of 200 runs that is a 42.5% reliability, and for 0.8 online probability gave 60.5% reliability and finally a reliability of 79.5% for a online probability of 0.9. The result from this test can be seen in Table II.

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

TABLE II

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

C. Result conclusion

The results from the simulation and the prototype have been compared with the probabilistic model. Only the test case with RS(3,1) was done for both tests, therefore these are the test results which will be compared. The results can be seen in Table III.

The results from Table III is shown in the graph on Fig. 13. From this graph it is clear that the simulation test verifies the probabilistic model and the results from the prototype test are lower. The dashed line on the graph is the non-cooperative model and it can be seen that the probabilistic model performs better than this model.

V. Discussion

The results of the tests have shown that the methods have proven to be suitable for the chosen scenario. The results can be seen in Fig. 13. The results from the simulation verifies the probabilistic system model and matches the results, there is only a small deviation ($\pm 0.6\%$). This deviation is most likely due to the random number generator in a PC which is not truly random, but only pseudo random [3]. The results from the prototype test shows a deviation from the system model and simulation, which must be considered acceptable since the prototype is just a real life proof of the model. The deviation is mainly due to a MAC protocol not suited for the scenario and a shared wireless medium. Even though collision avoidance mechanisms are implemented, it still needs fine tuning.

Online probability	**Probabilistic model [%]**	**Simulation [%]**	**Prototype [%]**
0.50	50	49.4	42.5
0.80	90	90.3	60.5
0.90	97.5	96.9	79.5

TABLE III

The results from the tests, where the reliability for probabilistic model, simulation and prototype is given for three distinct online probabilities.

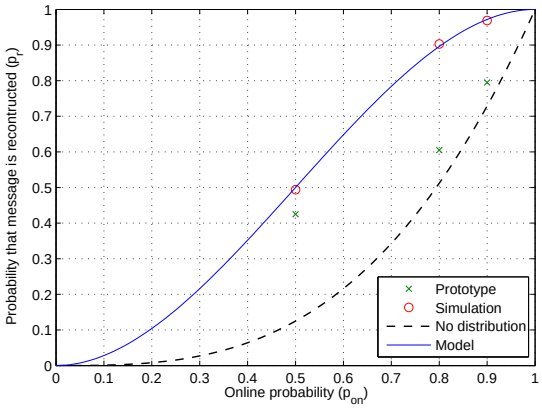

Fig. 13. The system reliability as a function of online probability for Reed-Solomon coding with two data devices and one redundancy device in comparison to no distribution

Fig. 13 and 14 shows that using the cooperative method using Reed-Solomon is better than the non-cooperative case. For example in Reed-Solomon a 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. A 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 %.

From Fig. 14 it can, as well, be seen that the system reliability is higher than the online probability of each mote, for large online probability. Furthermore it is seen that the Reed-Solomon coding scheme is able to maintain the same level of reliability, when the online probability is decreased and the number of motes in the system is increased.

VI. Conclusion

The objective of this project was to propose a reliable solution that enables distribution of data to a gateway in a partially wireless connected sensor network, in comparison to

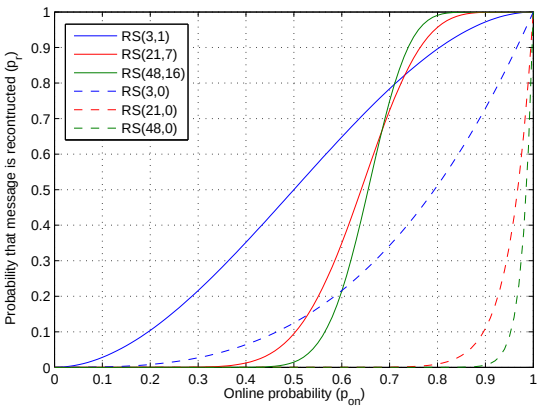

Fig. 14. Each solid line represents Reed-Solomon coding with a ratio of 3:1. RS(3,1) as an example shows Reed-Solomon coding with three devices, where one acts as redundancy device. The corresponding dashed lines show no distribution (ND) models.

a non-cooperative system where no distribution is performed. To fulfill this, the Reed-Solomon coding scheme has been selected to enable the gateway to recover data with a high reliability.

A probabilistic system model has been derived to find a relation between the online probability of each mote and the reliability of the system. This system model has been validated through a simulation and a prototype has been developed to test the model in a real life scenario. The result shows that the proposed solution has an enhanced performance compared to the non-cooperative case.

A. Future perspectives

The models proposed in this paper are static in the sense of the distribution phase, because all motes need to be online before the distribution can finish. In this section ideas to improve the model are proposed to make the model act dynamical in the distribution phase.

1) Distribution model: As stated earlier in this paper, the distribution is performed to all motes in the cluster. This results in extended online periods for each mote because it must wait for all motes to become online. For a small and predefined system this approach is suitable, but for a large and uncontrollable system it might take a long time to distribute if it is possible at all because a failure in one mote can result in a single point of failure. Instead of having such a static system where each mote is dependent on the other motes to be able to distribute its data a more dynamical approach could be introduced.

2) Search for available motes before distribution: Instead of waiting for motes to become online the distributing mote can perform a neighbor discovery to search for available motes and then distribute to those who are available. This will ensure that the distribution is performed much faster than by waiting for others to be online, but on the other hand the number of motes who will have a part of the message will not be as high as in the static approach. For a large scale network it can be assumed that a high percentage related to the online probability will have a part of the message. The implementation will also be better for a scalable network because the motes do not need to know how many neighbors it has in the cluster.

3) Multi hop distribution: The proposed model in this project is distributing the packets by making one-hop connections in the network. Another approach of distributing the packets could be to introduce multi-hop connections to let other motes in the system route the packets to their destination e.g. by using the epidemic algorithm [4]. By doing this, motes not in range of each other might still be able to communicate because some motes in between them can be used to convey the packets.

4) Poisson distribution: The binomial distribution is used to calculate the probabilistic model but it is limited to small n and m due to computation precision. If the model should show how the reliability is for high m and n the Poisson distribution must be used.

REFERENCES

[1] Ben Krøyer Frank Fitzek. *Mobile devices - openSensor*. `http://mobiledevices.kom.aau.dk/opensensor/`, 2007.
[2] Anders Grauballe and Mikkel Gade Jensen. Mac protocol for wireless sensor networks. University project report, 2007.
[3] Mads Haahr. *Introduction to Randomness and Random Numbers*. `http://www.random.org/randomness/`, 2007.
[4] Tom Daniel Hollerung. *Epidemic Algorithm*. `http://wwwcs.uni-paderborn.de/cs/ag-madh/WWW/Teaching/2004SS/AlgInterne%t`, 2004.
[5] Robert H. MarelosZaragoza. *Art of Error Correction Coding*. John Wiley and Sons Inc., 2th edition, 2006. ISBN: 0-470-01558-2.
[6] Jens Dalsgaard Nielsen. *kernels*. `http://www.control.aau.dk/~jdn/kernels/`, 2007.
[7] J. S. Plank. A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems. *Software – Practice & Experience*, 27(9):995–1012, September 1997.
[8] Sheldon M. Ross. *Introduction to Probability and Statistics for Engineers and Scientists*. Elsevier Academic Press, third edition, 2004. ISBN: 0-12-598057-4.