Fault Recovery in Single-Hop Sensor Networks

S. Chessa [*,1,2] and P. Maestrini [1,2]

[1] Dipartimento di Informatica, Università di Pisa, Corso Italia 40, Pisa, Italy.

[2] Istituto per la Scienza e Tecnologia dell'Informazione – CNR, Area della Ricerca di Pisa, via Moruzzi 1, Pisa, Italy.

Abstract

Wireless ad hoc Sensor networks are gaining increasing attention as they are suitable to monitor a wide variety of environments, where the deployment of classical wired communication infrastructure is neither economical, nor possible. The network collects environmental information which, in turn, is sent to a sink node (accessible to external users) when the connection with the sink node is available. If a sensor fails when the connection with the sink node is unavailable, all data sensed by this sensor are definitively lost. Neither the classical approaches to fault recovery nor the recently proposed approaches for the recovery in mobile computing are appropriate under this scenario, because they would require that the sensors be equipped with stable storage and/or a permanent connection with the sink node. Such requirement can hardly be met in sensor networks. This paper introduces a fault recovery scheme for single hop wireless sensor networks, where the network is able to operate independently of the sink node for long periods of time. The proposed scheme does neither require that the sensors be equipped with stable storage nor that they are permanently connected to the sink node. Rather, the sensors cooperate to maintain redundant information in their memories in order to be able to recover the lost information after a failure, and to distribute the recovered information among the memories of the surviving sensors. The proposed scheme minimizes the memory overhead.

Categories and subject descriptors: *D.4.5 [**Operating Systems**]: Reliability* – Fault-Tolerance, Backup Procedures, *C.2.4 [**Computer Communication Networks**]: Distributed Systems,* Distributed databases, *C.2.1 [**Computer Communication Networks**]: Network Architecture and Design,* Distributed networks, *E.4 [**Coding and Information Theory**]:* Error control codes.

1 Introduction

Wireless ad hoc networks [1] are gaining increasing attention due to their suitability to deployment in remote or hostile environments. Sensor networks deserve special interest, as they are suitable to monitor environments where the deployment of wired communication infrastructure is neither economical, nor possible.

Sensors are equipped with sensing, processing and wireless communication devices, and they cooperate in order to collect information from the surrounding environment. The collected information is transmitted to an external sink node at discrete times, when a connection with the sink node is available. The sink node, which could be either a base or a mobile station, is accessed by the external operators to retrieve the information gathered by the network.

The connection between the sink node and the network may be unavailable for long periods of time for several reasons. For example, if the network is deployed in an hostile environment, it should be able to operate autonomously for (possibly) long periods, until the sink node establishes a connection with the network in order to gather information from the network.

More specifically, consider an application in which a wireless sensor network is deployed in a desert or a different planet for the purpose of collecting environmental data, and the sink node is carried by an airplane or by a satellite flying over the area where the network is deployed. The connection is thus provided only for short periods of time. It should also be considered that communication between the sensors and the sink node may require relatively high power and may be subject to strong interference and noise.

In most applications, the sensors are extremely compact and rely on on-board batteries for energy supply. Once a sensor depletes its battery, it becomes unreachable from the other sensors in the network, and may be considered faulty at any effect. This means that the lifetime of the network is limited. Several techniques have been proposed to extend the network lifetime, either improving the energy efficiency of the sensors [2], or exploiting energy efficient cooperation strategies [3,4,5]. In general, if sensor battery replacement is not

possible, the network lifetime cannot be extended over a certain limit. An upper bound to this limit has been recently derived in [6]. Scenarios in which battery replacement is feasible have been studied in [7].

If a sensor fails when the connection with the sink node is unavailable, its sensed data may be definitively lost, unless the sensor is equipped with permanent storage (which may be prevented by the size and energy constraints to which the sensors are subject) or it is reachable by the external operators (which is prevented in the case of hostile or remote environments). Under this scenario, in order to avoid data loss, some recovery strategies need to be introduced.

The usual recovery approaches based on checkpointing and/or message logging cannot be used in this case, since the failed sensors cannot restart, and the sensors are not equipped with permanent storage. Recovery techniques suitable for mobile networks where mobile hosts do not have permanent storage have been recently proposed [8,9,10,11,12]. Such techniques exploit base stations to store message logs and/or checkpoints, and require almost permanent connection (that is, sporadic data loss are tolerated) of the mobiles with the base station.

However almost permanent connection cannot be guaranteed in most applications of sensor networks. In this environment, approaches where the recovery is coordinated by the sink node should be discarded, since they require efficient and almost permanent connection with the network. Considering that sensors cooperate to achieve a common sensing task, and the main focus of the recovery should be avoiding data losses, we propose an approach in which the sensors cooperate to maintain redundant information in their memories in order to be able to recover the lost information after a failure. More specifically, the sensors organize their memory in order to store both sensed and redundant data. After a sensor failure redundant data are used to recover the data sensed by the failed sensor. Once recovered, such data are distributed in the memories of the surviving sensors and, when the connection with the sink node becomes available, they are sent to the sink node along with the data sensed by the surviving sensors.

The proposed recovery strategy requires additional memory resources and consumes energy due to communication overhead. A recovery session roughly requires the same amount of messages as it would be required if message logs had been stored in the base station, however messages exchanged for the purpose of recovery in the proposed approach involve communication among sensors, which is much less energy demanding than the communication with a sink node carried by an airplane or a satellite.

Since the memory is also a resource subject to stringent constraints, the redundancy scheme should minimize the amount of memory required to store redundant data. For this reason we propose a redundancy scheme reminiscent of the bit striping technique of RAID systems [13], with the difference that, in this case, the sensors do not restart after a failure, and the redundant data should be restructured.

The rest of the paper is organized as follows. Section 2 introduces the system model, Section 3 describes the proposed redundancy management, and Section 4 introduces the recovery scheme. Section 5 analyzes the memory and the communication overheads, and Section 6 draws some conclusions.

2 Basic Assumptions and System Model

We consider a set of n_0 sensors, each of which is initially assigned a unique ID, ranging from 0 to n_0–1. A sensor can be in one of two states: faulty or non-faulty. Sensors may fail for several reasons, for example because of hardware faults, unrecoverable system crashes or battery depletion. In this paper we focus on *permanent* faults, that is, a sensor cannot toggle between the two states. This means that, once failed, a sensor cannot communicate with the other sensors any longer. In the following, we will refer to non-faulty sensors as *active* sensors.

We focus on the case of single faults, that is, we assume that no new faults occur during the time elapsed between a failure and the end of the recovery procedure. This assumption is realistic if the mean time between failures in the network is much larger than the fault detection and recovery execution time. In principle, failures may be detected when the sensors fail to reply to messages during the normal protocol execution. More sophisticated techniques can be used to reduce the elapsed time between a failure and its detection, for example, sensors could periodically send “I’m Alive” messages [15]. In the following we assume that sensors are able to detect failures, disregarding any detail about the fault detection protocol.

Each active sensor is assigned a unique temporary ID (TID), which initially corresponds to its ID. At a given time t, the TID of the sensor with ID i is determined as i–|{sensors with ID $<i$ failed within time t }|. It

follows that the TID, which changes after each failure, is different for each active sensor, and it is in the range $N=[0,n-1]$, where $n \leq n_0$ is the number of active sensors at time t. Hereafter, referring to a given time, notation u_i will denote a sensor with TID i. After a failure, during the recovery process, we will keep notation u_j to denote sensor with TID j computed before the failure, and notation $u_{[i]}$ to denote the sensor with TID i computed after the failure.

Active sensors communicate with each other via radio transceivers and they use a MAC protocol to solve contentions over the wireless channel. We assume that the communications are reliable and that the network is single-hop, that is, every transmission originated by a sensor reaches every other sensor in the network. This assumption deserves some comments. In general the transmission are unreliable, that is, messages may not reach their destination. This problem can be dealt with by managing appropriate timeouts and message acknowledgements and retransmissions. Since the recovery protocol described in this paper can be easily modified in order to take into account unreliable communications, for the sake of simplicity we will omit in the rest of the paper all the details related to unreliable communications.

Sensors are equipped with a limited amount s of memory used to store both the sensed and redundant information. Initially the entire content of this memory is reset to zero. When the connection with the sink node is not available, the sensors store the sensed data in their memories. When the connection with the sink node becomes available, the sensors send all the sensed data to the sink node. After that they reset their memory to start again the sensing task. We assume that this procedure is synchronized, that is, all the sensors provide sensed data to the sink node and then they reset their memories.

Since faulty sensors are unable to communicate, the content of their memories is inaccessible to the active sensors. When the active sensors detect a new failure, they execute a distributed recovery protocol to recover the sensed data of the failed sensor, and to distribute the recovered information to the active sensors. If the free memory of the sensors is not sufficient to store recovered data, the recovery process is not started and the content of the failed sensor is definitively lost. The recovered data will be sent to the sink node along with the sensed data when the connection with the sink node will be available again.

3 A Redundancy Scheme for Single-Hop Sensor Networks

3.1 Preliminary Definitions

The sensors memory is organized into s locations, each containing a data block. The set of data blocks is in turn partitioned into two areas, namely *data memory* and *redundant memory*. The former area includes locations in $M=[0,m-1]$ (with $m<s$) and it is used to store sensed data and data recovered from failures of other sensors. The latter area includes r locations (with $r=s-m$) in $[m,s-1]$ and it is used to store redundant data used for recovery (Figure 1). Given $l \in [0,s-1]$, and $j \in N$, we denote with $d_{l,j}$ the block contained in location l of the memory of sensor u_j. When referring to the redundant memory we also use notation $r_{x,j}$ (with $x \in R$, $R=[0,r-1]$) to denote the block contained in location x of the redundant memory of sensor u_j, that is, $r_{x,j}=d_{m+x,j}$.

We also introduce the following notations. Given a pair $p=\langle x,y \rangle$ we denote by $p|_1$ ($p|_2$) the first (second) element of p, that is, $p|_1=x$ ($p|_2=y$). Given two b-bit vectors $v=\langle v_0,\ldots,v_{b-1} \rangle$ and $w=\langle w_0,\ldots,w_{b-1} \rangle$, the operation $v \oplus w$ corresponds to the bit-wise XOR between the bits in v and the corresponding bits in w, that is, $v \oplus w=\langle v_0 \oplus w_0,\ldots, v_{b-1} \oplus w_{b-1} \rangle$.

3.2 Redundancy Management

In the proposed scheme, each sensor uses its redundant memory to store the parity of $n-1$ data memory locations belonging to different sensors.

The core of the redundancy management scheme consists in a set of functions $f : N \times M \mapsto N \times R$ and $g_k : N \times R \mapsto N \times M$ (with $k \in [0,n-2]$). Functions f takes as parameter a pair $\langle j,l \rangle$ denoting location l in the data memory of a sensor u_j, and returns a pair $\langle i,x \rangle$ such that $d_{l,j}$ contributes to the parity stored by u_i in location $r_{x,i}$ (that is, $\langle i,x \rangle$ *covers* $\langle j,l \rangle$). It maps up to $n-1$ different pairs of $N \times M$ on a unique pair of $N \times R$.

Each function g_k with $k \in [0,n-2]$ is one-to-one; for each k g_k takes as parameter a pair $\langle i, x \rangle$ denoting a redundant memory location $r_{x,i}$ in sensor u_i and returns a pair $\langle j_k, l_k \rangle$ such that $\langle i, x \rangle$ covers $\langle j_k, l_k \rangle$. If the

pair $<i, x>$ covers $h<n-1$ different pairs in $N \times M$ then only the functions g_k with $k<h$ are defined, while the remaining $n-1-h$ are undefined.

Formally, functions f and g_k are such that

$$g_k(f(j,l))=<j,l> \text{ for a unique } k \in [0,n-2],\ k \text{ integer.} \qquad (1)$$

Hereafter, if functions f and g_k ($k \in [0,n-2]$) satisfy the above defined property, we say that function f is invertible by the family of functions g_k, called the inverse family. We also say that $d_{l,j}$ with $l<m$ is *covered* by $r_{x,i}$ in the redundant memory of u_i if $<i,x>$ covers $<j,l>$.

At a given time during the lifetime of the network, sensor u_i stores in location x of its redundant memory the value

$$r_{x,i} = \bigoplus_{k=0}^{n-2} d_{l_k, j_k} \qquad (2)$$

where $<j_k, l_k>=g_k(i,x)$ for each $k \in [0,n-2]$. Letting k' be such that $<j,l>=g_{k'}(i,x)$ and inverting Equation 2, the content of location l in the data memory of sensor u_j can be written as

$$d_{l,j} = r_{i,x} \oplus \left(\bigoplus_{\substack{k=0\ldots n-2, \\ k \neq k'}} d_{l_k, j_k} \right) \qquad (3)$$

where $<i,x>=f(j,l)$, and $<j_k, l_k>=g_k(i,x)$ for each $k=0,\ldots,n-2$. It is immediate that if u_j fails, the contents of the data memory of u_j can be recovered using Equation 3 only if the right hand side of the equation is not dependent on the contents of the memory of u_j itself. For this reason we further restrict consideration to functions f and g_k satisfying the following conditions:

a) $<i,x>=f(j,l)$ implies that $i \neq j$;

b) $f(j,l_1) \neq f(j,l_2)$ iff $l_1 \neq l_2$;

the following theorem shows that the recovery is possible iff both a) and b) are met.

Theorem 1. Let function f be invertible by the inverse family g_k ($k \in [0,n-2]$), and let $j \in N$. Then, if u_j fails, any $d_{l,j}$ with $l \in M$ is recoverable using Equation 3 iff f and g_k satisfy conditions a) and b).

Proof. Let us assume that u_j has failed, and let $l \in M$ be any location in the data memory of u_j.

We firstly show that if functions f and g_k satisfy conditions a) and b) then $d_{l,j}$ can be recovered by exploiting the content of the memory of the active sensors. Let $<i,x>=f(j,l)$, by requirement (1) we have that there exists a unique $k' \in [0,n-2]$, k' integer, such that $<j,l>=g_{k'}(i,x)$, and by Equation 3 $d_{l,j}$ can be written as a function of $r_{x,i}$ and d_{l_k, j_k} with $k \in [0,n-2]$, $k \neq k'$ and $<j_k, l_k>=g_k(i,x)$.

Let now assume by contradiction that the right hand side of Equation 3 is dependent on data stored in u_j, that is, either $r_{x,i}$ or some d_{l_k, j_k} belong to u_j. From $<i,x>=f(j,l)$, by Condition a) we have that $i \neq j$, from which we conclude that $r_{x,i}$ does not belong to u_j. Hence there must exists an integer $h \neq k'$, $h \in [0,n-2]$ such that d_{l_h, j_h} belong to u_j, that is, $j_h=j$ and by hypothesis $<j_h, l_h>=g_h(i,x)$. Since by requirement (1) k' is the unique integer such that $<j,l>=g_{k'}(i,x)$, from $h \neq k'$ follows that $g_h(i,x) \neq g_{k'}(i,x)$, that is, $<j_h, l_h> \neq <j,l>$, and then $l_h \neq l$. Hence by condition b) we have $f(j_h,l_h) \neq f(j,l)$, and then $f(j_h,l_h) \neq <i,x>$, thus contradicting $<j_h, l_h>=g_h(i,x)$. Then must be $j_h \neq j$ for any $h \in [0,n-2]$, $h \neq k'$.

In conclusion, since $r_{x,i}$ and any d_{l_k, j_k} with $k \neq k'$ do not belong to u_j, they are all available after the failure of u_j, and then $d_{l,j}$ can be recovered by means of Equation 3.

We now show that if $d_{l,j}$ is recoverable from the failure of u_j by means of Equation 3 then f and g_k must satisfy conditions a) and b).

Let us assume that Condition a) does not hold, hence there exists a pair $<j,l>$ such that $f(j,l)=<j,x>$. In this case, after the failure of u_j, the recovery of data $d_{l,j}$ by means of Equation 3 would require the value of $r_{x,j}$ which is however not available because u_j has failed, thus leading to a contradiction.

If Condition b) does not hold, then there exist $j \in N$ and $l,l' \in M$ such that $l \neq l'$ and $f(j,l)=f(j,l')$. If sensor u_j fails, then Equation 3 would require data $d_{l,j}$ to recover $d_{l',j}$ and vice versa, hence the recovery of $d_{l,j}$ and $d_{l',j}$ would not be possible, leading to a contradiction.

We conclude that f and g_k satisfy the conditions a) and b). □

We now define functions f and g_k ($k \in [0,n-2]$) as follows:

A) given $j \in N$ and $l \in M$, $f(j,l) = \langle i,x \rangle$ with $i = (j+1+l \bmod (n-1)) \bmod n$ and $x = \left\lfloor \frac{l}{n-1} \right\rfloor$;

B) given $i \in N$, $x \in R$, and $k \in [0,n-2]$, $g_k(i,x) = \begin{cases} \langle j_k, l_k \rangle & \text{if } l_k < m \\ \text{undefined} & \text{otherwise} \end{cases}$ where $j_k = (i-k-1) \bmod n$ and $l_k = x(n-1)+k$.

According to A) and B), the relationship between the sensors data memory locations and sensors storing the respective parities is shown in Figure 2. From A) it is also immediate that $x \leq \left\lfloor \frac{m-1}{n-1} \right\rfloor$, from which it is seen that the amount of redundant blocks required by the proposed redundancy scheme is

$$r = \left\lfloor \frac{m-1}{n-1} + 1 \right\rfloor \qquad (4)$$

We now show that recovery from a single failure by means of functions f and g_k defined in A) and B) is always possible, that is, for any element $d_{l,j}$ in D there exists a unique $r_{x,i}$ such that $d_{l,j}$ is covered by $r_{x,i}$, and that each $r_{x,i}$ covers at most $n-1$ data memory locations belonging to different sensors. We first show that functions f and g_k satisfy Requirement 1.

Lemma 1. Function f defined in A) is invertible by the inverse family g_k ($k \in [0,n-2]$) defined in B). Also, $g_k(f(j,l))=\langle j,l \rangle$ with $k = l \bmod (n-1)$.

Proof. Let $j \in N$, $l \in M$, and $k = l \bmod (n-1)$, by definition of f we have that $f(j,l)=\langle i,x \rangle$ with $i = (j+1+l \bmod (n-1)) \bmod n$ and $x = \left\lfloor \frac{l}{n-1} \right\rfloor$, and by definition of g_k we have $g_k(i,x)=\langle j_k,l_k \rangle$ with $j_k = (i-k-1) \bmod n$ and $l_k = x(n-1)+k$, from which $j_k = ((j+1+l \bmod (n-1)) \bmod n - k - 1) \bmod n$ and $l_k = \left\lfloor \frac{l}{n-1} \right\rfloor (n-1)+k$. Hence follows that $j_k=j$ iff $(j+1+l \bmod (n-1)) \bmod n - k - 1 = \alpha n + j$ for some integer $\alpha \geq 0$, which holds iff $j+1+l \bmod (n-1) = \beta n + j + 1 + l \bmod (n-1)$ for some integer $\beta \geq \alpha$, which is true letting $\beta=0$.

Observing now that $l_k = \left\lfloor \frac{l}{n-1} \right\rfloor (n-1) + l \bmod (n-1) = \frac{1}{n-1}(l - l \bmod (n-1))(n-1) + l \bmod (n-1) = l$ we conclude that $j_k=j$ and $l_k=l$, and then $g_k(f(j,l))=\langle j,l \rangle$.

We show now that $k = l \bmod (n-1)$ is the unique integer in $[0,n-2]$ such that $g_k(f(j,l))=\langle j,l \rangle$. Let $\langle i,x \rangle = f(j,l)$ and assume by contradiction that there exists an integer $h \in [0,n-2]$, $h \neq k$ such that $g_h(i,x)=g_k(i,x)$. From B) must be $l = x(n-1)+k$ and $g_h(i,x)=\langle j_h,l_h \rangle$ with $l_h = x(n-1)+h$, from which $l \neq l_h$ and $g_h(i,x) \neq g_k(i,x)$. The thesis follows immediately. □

Theorem 2. Let $j \in N$ and f and g_k ($k \in [0,n-2]$) be as defined in A) and B). Then, if u_j fails, any $d_{l,j}$ with $l \in M$ is recoverable using Equation 3

Proof. By Lemma 1 the functions f and g_k defined in A) and B) satisfy Requirement 1, hence by Theorem 1 it is sufficient to show that they also satisfy conditions a) and b).

Condition a): $\langle i,x \rangle = f(j,l)$ implies that $i \neq j$. Let $j \in N$, $l \in M$, and $f(j,l) = \langle i,x \rangle$. By A) we have that $i = (j+1+l \bmod (n-1)) \bmod n$; assuming by contradiction that $i=j$ we have that $i + \alpha n = i + 1 + l \bmod (n-1)$ for some integer α, from which $\alpha n - 1 = l \bmod (n-1)$ which is a contradiction. Hence a) is satisfied.

Condition b): $f(j,l_1) \neq f(j,l_2)$ iff $l_1 \neq l_2$. It is immediate that $f(j,l_1) \neq f(j,l_2)$ implies $l_1 \neq l_2$. Conversely, let $j \in N$, $l_1 \in M$, $l_2 \in M$, and let $f(j,l_1)=\langle i_1,x_1 \rangle$ and $f(j,l_2)=\langle i_2,x_2 \rangle$. Let us assume that $l_1 \neq l_2$, and assume by contradiction that $f(j,l_1) = f(j,l_2)$. Then, from $i_1=i_2$ then must be $(j+1+l_1 \bmod (n-1)) - (j+1+l_2 \bmod (n-1)) = \alpha n$ for some

integer α, which implies that $l_1 \bmod(n-1) - l_2 \bmod(n-1) = \alpha n$, from which α=0. Hence it results that $l_1 - l_2 = \beta(n-1)$ for some $\beta \neq 0$, from which $\left\lfloor \frac{l_1}{n-1} \right\rfloor \neq \left\lfloor \frac{l_2}{n-1} \right\rfloor$ and then $x_1 \neq x_2$ which is a contradiction. Hence it follows that $l_1 \neq l_2$ iff $f(j,l_1) \neq f(j,l_2)$.

□

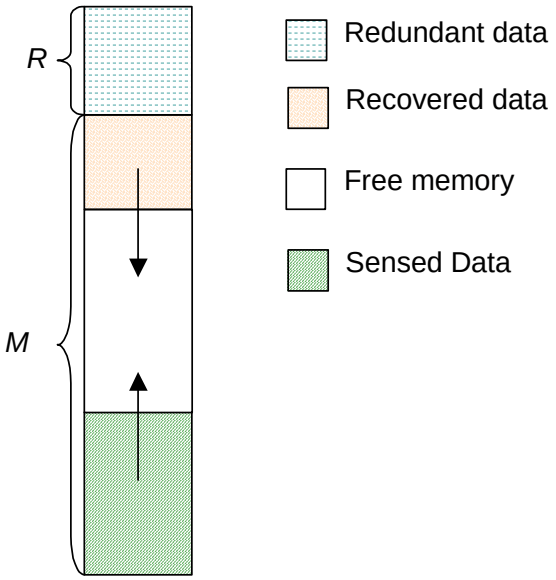

Figure 1. Memory management in a sensor.

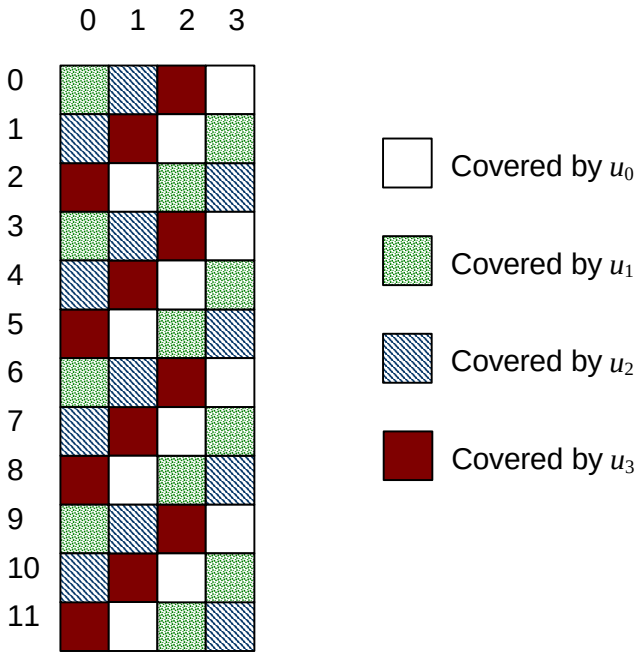

Figure 2. Relationship between the sensor data memory locations and the sensors which cover them in the case of n=4 and m=12.

3.3 The Recovery Protocol.

During the network lifetime, each sensor senses and stores data in its data memory. When sensor u_j wants to update a data memory location l, it computes $f(j,l)$ and it sends an *update request* message to sensor u_i with $i=f(j,l)|_1$. The update request message contains a pair $<d,l>$ containing the data d and the location l in the data memory where the sensed data will be stored. When u_i receives the update request message $<d,l>$, it updates its redundant memory in location $x=f(j,l)|_2$, letting $r_{x,i}=r_{x,i} \oplus d$. Each other sensor u_k with $k \neq i$ stop hearing the update request message as soon as it realizes that it is not the destination of the message in order to save energy [2].

When the active sensors detect the failure of a sensor u_j they start the recovery process provided they have enough free memory to store the recovered data. Since each active sensor recovers data of u_j covered by its redundant memory, each active sensor recovers up to r data locations of u_j. The protocol recovers both data sensed by u_j and data recovered by u_j from previous failures.

The recovery protocol described below is executed only if the failure occurs when the network is disconnected from the sink node. If this is not the case then the surviving sensors do not execute the recovery protocol, but they provide also their redundant memories to the sink node, which, in turn, recovers the data memory of the failed sensor.

The recovery of the data locations of u_j is executed by exploiting Equation 3. In particular, for each $k \in N$, $k \neq j$, sensor u_k sends the entire content of its data memory to the other sensors by sending *recovery messages*, where each recovery message contains a data block. Letting $<i,x>=f(k,l)$, the data block $d_{l,k}$ is sent by u_k to u_i, which, in turn, updates its redundant memory in location x letting $r_{x,i}=r_{x,i} \oplus d_{l,k}$. It is immediate that, at the end of this process, location x of the redundant memory of sensor u_i will store $r_{x,i} \oplus \left(\bigoplus_{k=0,n-2,k\neq k'} d_{l_k,j_k} \right)$ with $<j_k, l_k>=g_k(i,x)$ and $<j, l>=g_{k'}(i,x)$, which, by Equation 3, equals $d_{l,j}$. This also means that, during the recovery, each sensor processes up to $r \cdot (n-2)$ recovery messages. After the recovery the remaining active sensors compute the new TID's and they proceed with the normal sensing task. The memory content of u_j is spread among the remaining active sensors, which store these data in order to provide the sink node a snapshot of the memory of u_j before its failure. Since the recovered data will not be used by the surviving sensors it can be compressed in order to save precious sensor memory.

After the data memory of u_j has been recovered, the active sensors execute a procedure to rebuild the redundant memory. In principle this requires that the sensors flush again the content of their data memories, but in practice it can be achieved during the recovery by exploiting the recovery messages. For this reason, before starting the recovery, each active sensor computes the new amount r' of redundant memory necessary to cover the data memory of the remaining active sensors after the failure of u_j, and it shifts the current recovered and redundant data to reserve r' locations in the top of the memory (Figure 3). Then it exploits the recovery messages sent during the recovery to determine the new content of the redundant memory. To this purpose a recovery message containing data $d_{l,k}$ will be processed by the sensors u_i and $u_{[h]}$, where $i=f(k,l)|_1$ with f computed referring to the TIDs before the failure of u_j, and $h=f(k,l)|_1$ with f computed referring to the new TIDs after the failure of u_j. However, since the data recovered from previous failure has been shifted to new locations, the sensors will send recovery messages in the form of triples $<d_{l,k},l_n,l_o>$, where l_n is the new memory location of the data and l_o is the memory location of the data before the shift. Observe that $l_n=l_o$ if the data block is a sensed data, and $l_n=l_o-r'$ if the data block is a data recovered from previous failures. The original location is used to recover the data of the failed sensor, and the new location is used to compute the new content of the redundant memory.

The recovery blocks reserved in each sensor to store the recovered data of u_j were free blocks before the recovery, and, being their content zero, they were not necessary to recover the data of u_j. However, when the recovery is complete those blocks are not covered by the redundant memory of the surviving sensors. For this reason, after the recovery, the active sensors send update request messages containing those memory blocks.

In many cases the recovery communication overhead can be significantly reduced by avoiding to send free memory blocks (i.e. whose content is zero), as zero blocks are not necessary to data recovery based on Equation 3. This is particularly effective if the sensors memories are not full at the time of recovery. Furthermore the communication overhead due to channel contention can be avoided by executing the recovery with high priority, and by assigning the channel to the sensors based on their TID.

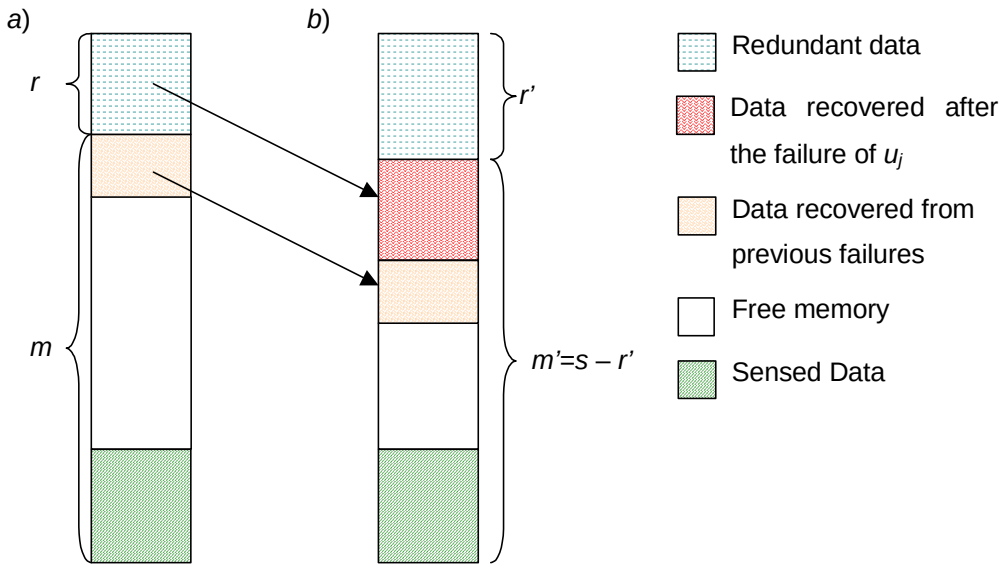

Figure 3. Memory management in u_i: *a*) before failure of u_j, and *b*) after failure of u_j.

4 Analysis and Discussion

4.1 Memory Overhead

From Equation 4, the amount r of redundant memory per sensor should be such that $\frac{m-1}{n-1} \le r \le \frac{m+n-2}{n-1}$, where n is the number of active sensors, m is the number of blocks reserved for the data memory and $s=m+r$ is the amount of memory blocks available to the sensors, from which follows that

$$\frac{s-1}{n} \le r \le \frac{s+n-2}{n} \qquad (5)$$

The memory overhead is defined as the ratio r/s (where s is the amount of memory blocks available to the sensors and r is the number of redundant blocks). Since the proposed redundancy scheme is a variation of the bit striping technique [13], the memory overhead should be about $1/n$ memory locations per sensor in a network with n sensors. Precise memory overhead is derived from Equation 5 as

$$\frac{1}{n} - \frac{1}{s \cdot n} \le \frac{r}{s} \le \frac{1}{n} + \frac{1}{s} - \frac{2}{s \cdot n} \qquad (6)$$

We now evaluate the amount of memory per sensor required to store redundant and recovered data after h sensor failures, under the worst case assumption that the recovered data is stored not compressed. Let n_0 be the initial number of sensors, s be the amount of memory available to the sensors, and let $m(n_0,s,h)$ be the amount of memory reserved to store redundant and recovered data in a sensor after h sensor failures. Letting $r(n,s)$ be the amount of redundant memory as a function of the number of active sensors and of their memory, from Equation 5 we have that $m(n_0,s,h)$ is:

$$m(n_0,s,h) = \sum_{p=0}^{h} r(n_0-p,s) \le \sum_{p=0}^{h} \frac{s+n_0-p-2}{n_0-p} = h+1+(s-2)\sum_{p=n_0-h}^{n_0} \frac{1}{p}$$

and the fraction of memory reserved to store redundant and recovered data per sensor $\varphi(n_0,s,h)$ is

$$\varphi(n_0,s,h) = \frac{m(n_0,s,h)}{s} = \frac{h+1}{s} + \frac{s-2}{s} \sum_{p=n_0-h}^{n_0} \frac{1}{p}$$

Since in general $s>>h$, we obtain:

$$\varphi(n_0,s,h) \cong \sum_{p=n_0-h}^{n_0} \frac{1}{p} \qquad (7)$$

The latter expression, which approximate the fraction of memory dedicated to store redundant and recovered data in each sensor after h sensor failures, is depicted in Figure 4 for $n_0 \in [10,50]$ and $h \in [0,5]$. It is seen that $\varphi(n_0,s,h)$ increases more rapidly for small values of n_0, and that, if $n_0 >> h$ (for example when $n_0=30$ and $h \leq 5$) $\varphi(n_0,s,h)$ remains relatively small.

4.2 Recovery Overhead

The time required to complete the recovery procedure is dependent on the number of recovery/update messages, for this reason we evaluate the number of packets exchanged during the recovery under the worst-case assumption that the memory of the sensors is almost full and then each sensor sends up to m recovery/update messages. For the sake of simplicity we also assume that the size of the memory blocks fits a network packet.

Let b bytes be the block size and t bytes be the amount of memory available to the sensors (hence $s=t/b$). Recalling that $m=s-r$, from Equation 5, we have $m \leq \frac{s \cdot n - s + 1}{n}$, and then $\frac{s \cdot n - s + 1}{n}$ is an upper bound to the number of packets sent by each sensor. Therefore in the worst case the entire recovery process requires up to $pkt(n,t) = \frac{t \cdot n - t + b}{b}$ packets. Assuming blocks of size 2048 Bytes a plot of $pkt(n,t)$ for some values of n and t is shown in Figure 5, from which it is seen that the number of packets scales almost linearly with n and t.

The amount of messages sent during the recovery roughly corresponds to the number of messages required by the sensors to flush their memories to the sink node. However messages exchanged for the purpose of recovery involve communication among sensors, which is much less energy demanding than the communication with a sink node carried by an airplane or a satellite.

The time required to complete the recovery could be reduced by increasing redundancy. The simple solution of data memory duplication is not feasible as it leads to a great memory overhead. A simple and better solution consists in dividing the set of sensors into groups, where each group implement the proposed redundancy scheme independent of each other. Assuming groups of the same size, the memory overhead would be balanced, and the time to recovery would be reduced of a factor equal to the number of groups.

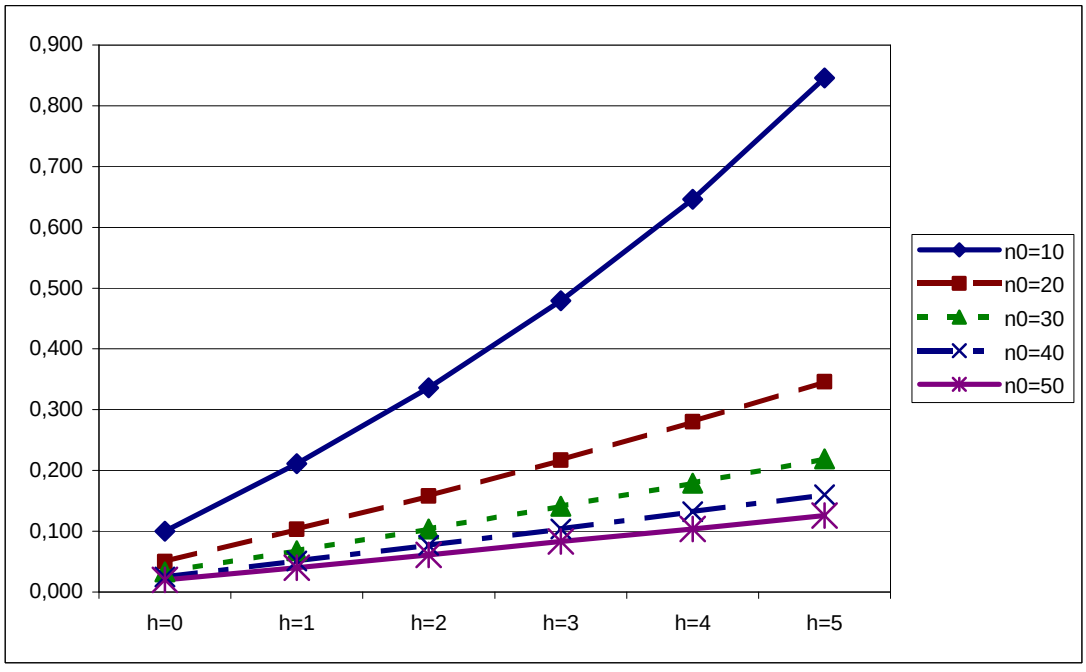

Figure 4. Plot of the approximate expression of $\varphi(n_0,s,h)$ for $n_0 \in [10,50]$ and $h \in [0,5]$.

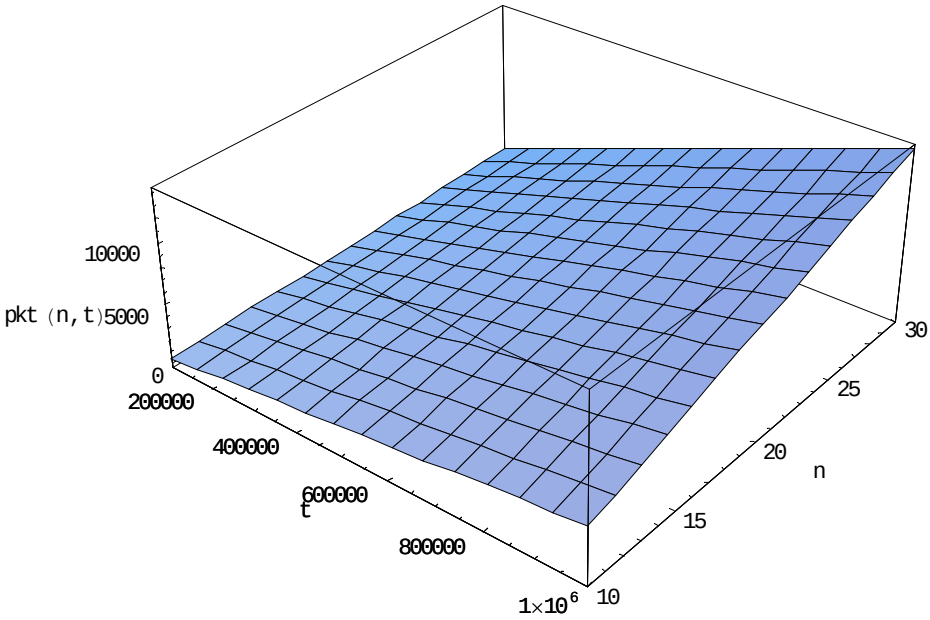

Figure 5. Plot of $pkt(n,t)$ for $n \in [10,30]$ and $t \in [10^5, 10^6]$.

5 Conclusions and Future Work

We have introduced a fault recovery scheme for single hop wireless sensor networks, where the network is able to operate independently of the sink node for long periods of time. The proposed scheme does neither require that the sensors be equipped with stable storage nor it relies on permanent connection with the sink node. Rather, the sensors keep redundant data in their memories in order to recover data lost after a sensor failure, and the recovered data is distributed in the memory of the surviving sensors to be sent to the sink node when it becomes available.

The redundancy scheme requires less energy than the communication of the network with the sink node and minimizes the memory overhead. It is shown that the memory overhead is mainly related to the number of sensors and quite independent of the memory size of the sensors. For example, this overhead is about 10% in a network with 10 sensors. Similarly, it is shown that the memory overhead after sensor failures is again related to the number of sensors and to the number of failures.

Future work includes recovery schemes for multi-hop networks, where sensors can be mobile, dealing also with transient faults.

6 References

[1] W. Diepstraten, G. Ennis, and P. Berlanger, "DFWMAC: Distributed Foundation Wireless Medium Access Control", IEEE Document P802.11-93/190 (November 1993).

[2] L. M. Feeney, M. Nilsson, "Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment", Proceedings of IEEE Infocom, Anchorage AK, April, 2001

[3] D. Estrin, R. Govindan, J. Heidemann and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks", *Proc. MOBICOM 99*, Seattle, WA, pp. 263-270, 1999.

[4] W.R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks", *Proc. MOBICOM 99*, Seattle, WA, pp. 174-185, 1999.

[5] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks", *Proc. MOBICOM 2000*, Boston, MA, pp. 56-67,

2000.

[6] M. Bhardwaj, T. Garnett, and A. P. Chandrakasan, “Upper Bounds on the Lifetime of Sensor Networks”, Proc. IEEE International Conference on Communications (ICC 2001), June 2001.

[7] P. Santi and S. Chessa, “Crash Faults Identification in Wireless Sensor Networks”, Computer Communications, 25 (14) (2002), pp. 1273-1282

[8] D. K. Pradhan, P. Krishna, and N. H. Vaidya, “Recovery in Mobile Environments: Design and Trade-off Analysis”, Proceedings of the 26th IEEE International Fault-Tolerant Computing Symposium, June 1996, pp. 16-25.

[9] R. Prakash and M. Singhal, “Low-Cost Checkpointing and Failure Recovery in Mobile Computing Systems”, IEEE Trans. On Parallel and Distributed Systems, pp.1035-1048, October 1996

[10] B. Yao, Kuo-Feng Ssu, W. K. Fuchs, “Message Logging in Mobile Computing”, Proceedings of the 29th Annual IEEE International Fault-Tolerant Computing Symposium, June 1999, pp. 294-301.

[11] B. Yao, W. K. Fuchs, “Proxy-based Recovery for Applications on Wireless Hand-held Devices”, Proceedings of the IEEE Symposium on Reliable Distributed Systems, October 2000, pp. 2-10.

[12] G. Krishnamurthi, S. Chessa, and A. K. Somani, "Fast Recovery from Database/Link Failures in Mobile Networks", Computer Communications, vol. 23 (5-6) 2000, pp. 561-574.

[13] D. Patterson, G. Gibson, and R. Katz, "A case for redundant arrays of inexpensive disks (RAID)," in Proceedings of ACM SIGMOD Conference on the Management of Data, pp. 109--116, 1988.

[14] Vinogradov, I., M., An Introduction to the Theory of Numbers, Pergamon Press, London & New York, 1955.

[15] T.D. Chandra, and S. Toueg, “Unreliable Failure Detectors for Reliable Distributed Systems”, *Journal of the ACM*, vol.43, n.2, pp.225-267, 1996.