# Probabilistic Checkpointing Protocol to Sensor Network Fault-Tolerant

**Faiza Titouna[1] , Chafiq Titouna[2], Salem Benferhat[3]**

**[1] Computer Science Department, El Hadj Lakhdar University
Batna, Algeria**

**[2] Ecole Doctorale STIC, M'Sila University
M'Sila, Algeria**

**[3]CRIL-CNRS UMR 8081, Sciences Faculty Jean Perrin, Artois University
Lens, France**

**Abstract**
A wireless sensor network WSN is a collection of autonomous sensors nodes organized into a cooperative network. A sensor node transmits the data quantity to the sink. Indeed, a failed sink may abort the overall mission of the network. Due to their crucial functions, sinks must be designed and maintained to be robust enough in order to face trouble coming from the harsh environment. Thus, as a keystone of a WSN, a sink has to be provided with ability to recover from failures. In this paper, we propose a new protocol avoiding to the sink to be a central point of failure. First, we model a sensor node failure estimation problem through a causal network. Then, we show how the checkpointing process ensures the recovery of the network. This approach reduces both energy consumption and communication bandwidth requirements, and prolongs the lifetime of WSN. Interesting results are given by simulation.

***Keywords:*** *Wireless sensor network, Fault tolerance, Checkpoint/Restart, Bayesian network.*

## 1. Introduction

Wireless sensor nodes are exceptionally complex systems where a variety of components interact in a complex way. Furthermore, hundreds or maybe thousands of these nodes will form a distributed embedded network system that will handle a variety of sensing, actuating, communicating, signal processing, computation, and communication tasks. Wireless sensor networks will be often deployed as consumer electronic devices that will put significant constraints on the cost and therefore, quality of used components. More importantly, nodes operate under strict energy constraints that will make energy budget dedicated to testing and fault tolerance very limited.
Wireless sensor networks themselves are a new scientific and engineering field and it is not still quite clear as to what is the best way to address a particular problem. In this situation, it is also difficult to accurately predict the best way to treat fault tolerance within a particular wireless sensor network approach. If one wants to ensure fault tolerance during a considered processing, the goal is to design fault tolerant techniques that do not significantly increase the communication overhead. On the other hand, if the computation energy is significantly higher than the communication requirements, it is a good idea to support communication resources at one node with the computation resources at other nodes. It is preferable to develop fault tolerant sensor approaches that require little additional computation regardless of any additional communication requirements.
In many applications, wireless sensor networks will be operational in harsh environments and thus nodes are usually exposed to increase risk of malfunction and physical damage. Tolerating the failure of the sink is usually necessary in such applications in order to avoid the loss of important sensors' data.
We offer a solution to the sensor node failure problem. Our approach allows a recovery of the sensor network when a specific node which is the sink is failed. The sink constitutes the key of WSN and its failure implies the failure of the whole components of the system. This work structures the relationship between some relevant parameters as a Bayesian network. Upon the failure of the sink, a recovering path is created and a consistent global checkpoint is determined. Finally, we show how the checkpointing process ensures the recovery of the network.

The remainder of the paper is organized as follows. After starting with a brief introduction, the section 2 provides relevant related work. In section 3, we present the developed approach which emphasizes the fault tolerance of the sink. The proposed approach is evaluated by simulation results presented in section 4. The last section concludes this work.

## 2. Related Work

The fault tolerance in wireless sensor networks is studied in several works. In [3], the proposed protocol is based on artificial intelligence; the clusters are constructed to transmit the data to multiple Sinks to ensure autonomy and a tolerance for failures. In [7], two types of Sinks are used, one static and the other dynamic, that improve the lifetime of the network. In [6], the authors have proposed an approach that allows a node to run self-diagnostic based on the accelerometer measures that will determine if the node induces a malfunction of the equipment. Another approach [6], [9] analyzes the curve of battery discharge and the rate of actual discharge, the algorithm proposed may estimate the remaining time for the total depletion of the battery. Another type of works based on the hierarchical network topology where the cluster-head controls its nodes, and the Sink controls the cluster heads. So, the Sink and the cluster heads run constantly a "ping" to the nodes under their direct monitoring. If a node does not respond, it will be wrong [10]. Faults can be recovered independent of applications. For instance, CODA [11] uses two mechanisms to mitigate congestion. When congestion occurs, a hop-by-hop back pressure message is propagated to the sources. When aggregated data needs to be collected, in-network aggregation has been accepted as a standard way to reduce communication overhead by pushing part of the aggregation to some intermediate sensor nodes. SKETCH [30] uses a DAG instead of a tree for data delivery. Given that most nodes have multiple parents in a DAG, an individual link or node failure has limited effects [12]. In [13] authors have proposed a cluster-based recovery algorithm, which is energy-efficient and responsive to network topology changes due to sensor node failures. The proposed cluster head failure-recovery mechanism recovers the connectivity of the cluster in almost less than of the time taken by the fault-tolerant clustering proposed by Venkataraman. The Venkataraman algorithm is the latest approach towards fault detection and recovery in wireless sensor networks and proven to be more efficient than some existing related work. Venkataraman algorithm is more energy efficient in comparison with Gupta and Algorithm Greedy.

## 3. Adaptive Sink fault-tolerant (ASFT)

### 3.1 Model of the system

In this approach, all sensor nodes in the network are assumed to be homogeneous, therefore, they have the similar capabilities of sensing, processing and communication. These nodes are spread along a geographical area, and can send information packets among themselves. Hence, the aggregation and the broadcast functions are provided by a specific node of the network. The selected node, that must ensure these functions, must have the characteristic of accessing to the base station with a maximal energy. Such node is called the sink (S). Another category of sensor nodes which are located in the network are defined in the sensing region. These nodes capture physical phenomena such as level of humidity, temperature, etc. We denoted by $NS_1$, $NS_2$,...,$NS_n$ this category. The rest of nodes allow the transmission of packets as showed in the Figure1. The initial state of the network is equipped by an initial sink $S_1$, which ensures the role of collector and transmitter data to the system users. This task will trigger when the users' queries arrive or when the time θ1 is elapsed. In this case, the sink broadcasts the corresponding query ($Query_{Interest}$) in the network.

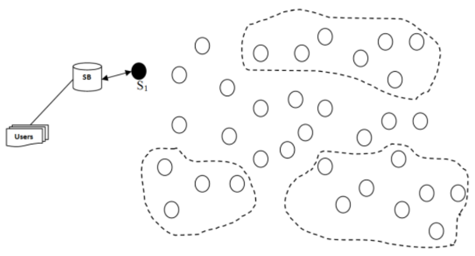

Fig.1 Example of a wireless sensor network

The sensor nodes receive all queries which are performed systematically and then transmitted back the captured data to the $S_1$. The selected sink collects raw data from the neighbouring sensing nodes, combines them by data fusion methods, and transmits the aggregated result back to the base station for a higher level processing. The Algorithm 1 presents the detailed behaviour of the initial sink $S_1$.
In this approach, we adopt two fault models which may be occurred at the sink, the lack of energy and an unexpected physical fault. At every moment of running the proposed protocol, we assume that every sensor node may know his level of energy denoted by $E$. Let be $E_{Threshold}$ the required minimal energy for the best performance of the sink and the best quality service. This fact allows to the nodes to expect and to detect instantly different faults.

**Alg1: Behaviour of initial sink $S_1$**

```
Begin
  // Request of initialisation of the network
    Broadcast (Initialisation_Query)
   Repeat
     Initialise($\theta_1$, Sensing_Data_Table)
     If Reception( Interest_ Query, SB) OR Timeout ($\theta_1$)  Then
          Distribute ( Interest _ Query, BROADCAST)
     If Reception ( Interest _Response, $NC_j$)) Then
          Adding( Interest_Response ,Sensing_Data_Table)
          Computing_fusion (Sensig_Data_Table)
          Broadcsat (Result_Computing, SB)
    Until (End of requests)
End
```

We present in the following sections the different process that composed the proposed protocol.

## 3.2 Recovery path creation process

We consider two principles metrics for creating the recovery path. The first consists of accessing to the base station and the second is the energy level. If the initial sink $S_1$ breakdown, becomes dysfunctional or the level energy consumption reaches the $E_{Threshold}$, at this time another sink, denoted by $S_2$, must take over. The choice of $S_2$ is depending on the metrics defined above. If we apply iteratively this process for N times, the last sink obtained will be denoted by $S_n$. The Figure 2 shows the process of creating the recovery path.

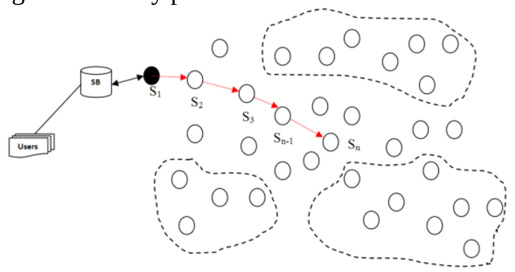

Fig.2 Recovery path creation process

### 3.2.1 Broadcast candidature query

The node $S_{j,j=1,n-1}$, sends the candidature query to its neighbourhood. At this time, the candidate node $C_N$, which has the possibility to access to the base station (in terms of minimal distance separating the node $S_j$ and the base station), sends a report $R_{Cn}$ that contains the energy level $E$ and the number of neighbours.
The recovery path process described above may be modelled as a tree (Fig3), where $S_1$ is considered as the root of the tree and the neighbours as the leaves.
The Algorithm 2 presents the detailed behaviour of the candidate sink $S_C$. All the reports are saved in the candidature table $T_{Cand}$ elaborated at the level of the node $S_j$.

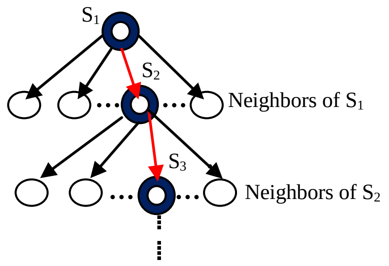

Fig.3 An example of initial recovery path tree

**Alg2: Behavior of Candidate_node $C_N$**

```
Begin
 If Reception (Candidature_Query, $S_j$)  Then
     $Acc_{SB}$ = True ;
     $N_{neig}$ = number_neighbors;
     $E_{Cn}$ = compute_energy() ;
     $R_{Cn}$= [$Acc_{SB}$, $N_{neig}$, $E_{Cn}$]
     Send ($R_{Cn}$, $S_j$)
End
```

### 3.2.2 Selection of the successor sink

Firstly, we use the Bayesian network for describing the model of the sink. The Figure 4 illustrates a simple typical Bayesian network. The nodes represent propositional variables of interest and the links represent causal independencies among the variables. The dependencies are quantified by conditional probabilities for each node given its parents in the network. The network supports the computation of the probabilities of any subset of variables given evidence about any other subset.

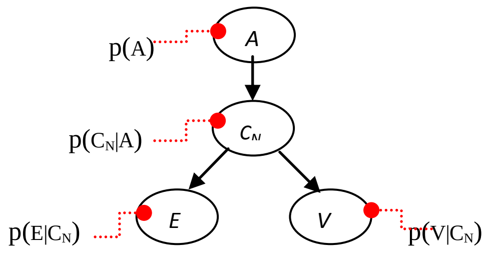

Fig.4 A Bayesian network as a sink model

The qualitative component of the network expresses that the candidate node ($C_N$) becomes a sink, whether it has the ability to access to the base station. So, if the candidate node is considered as the sink then it is more plausible that its level of energy ($E$) satisfies the relation $E > E_{Threshold}$ *and* the maximum of its neighbours ($V$) is reached. If the level of energy of the candidate node is lower than the

threshold $E_{Threshold}$, then we can say that it is less possible that the candidate node becomes the sink.
The quantitative component of the network is described by the different local conditional probability distributions associated with each node given its parents $p(X_i|par_{xi})$. For example, the local probability distribution associated with the node ($C_N$) of the network described in the Figure 4 is computed using the Bayes'rule [14]:

$$p(C_N|A) = \frac{p(A|C_N).p(C_N)}{p(A)} \qquad (1)$$

The joint probability distribution $P^J(X_1, \cdots, X_n)$ which encapsulates all the variables together is defined using the chain rule based on the product as follows [15]:

$$P^J(X_1, \cdots, X_n) = \prod_{i=1}^{n} p(X_i|par_{Xi}) \qquad (2)$$

Where $X_i$ represents the variable defined on the network and $par_{X_i}$ represents the parents of the node $X_i$.

Matching the eq.(2) on the Bayesian network described in the Figure 4, we obtain the following equation:

$$P^J(A,S,E,V) = p(A).p(C_N|A).p(E|C_N).p(V|C_N) \qquad (3)$$

Now, we can easily calculate the posterior probability under evidence. Let's compute the probability of that the candidate node ($C_N$) will be a sink if it has the possibility to access to the base station.

$$p^m(C_N = true|A = true) = \frac{\sum_{E,V} P^J(A=true, C_N=true)}{p(A=true)}$$
$$= \frac{\sum_{E,V} P^J(A=true, C_N=true, E=e, V=v)}{\sum_{C_N,E,V} P^J(A=true, C_N=c, E=e, V=v)} \qquad (4)$$

The first equality holds by the Bayes theorem and the second equality holds by the marginality of the probability. Given the conditional probability $p(C_N|A)$ which serves the role of a sink model and can be thought of in two ways. First, in building a sink model, the probability is constructed by fixing the value of A = a and then asking what probability density $p(C_N|A{=}a)$ on $C_N$ results. Conversely, when this sink model is used and observation is made, $C_N$ = c is fixed and a likelihood function $p(C_N = c|A)$ on (A) is inferred. The likelihood function, while not strictly a probability density, models the relative likelihood that different values of (A) gave rise to the observed value of $C_N$. The product of this likelihood with the prior, both defined on A, gives the posterior or observation update $p(A|C_N)$.

Thus, the node who has the highest probability value will be selected as the successor sink $S_{j+1}$. Therefore, two messages are sent to the successor node:

- The identification predecessor $S_{j-1}$ of $S_j$: this message is required for establishing the failure recovery of the node $S_j$.
- "You are the successor sink": this message is sent to the successor sink node.

After receiving the second message by the successor candidate node, this process is iteratively executed. Finally, we obtain a final path modelled as a causal network (see Figure 5).

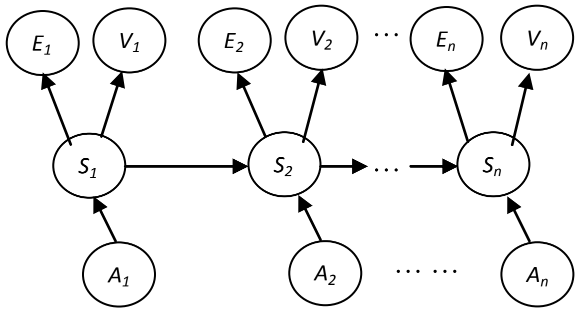

Fig.5 Bayesian network for the final description of the Sink path

This path reinforces the sink fault tolerance in the considered wireless sensor network. So, $S_n$ will be the last sink in the recovery path. The detailed procedure explaining the behaviour of $S_j$ is presented by the Algorithm 3.

**Alg3: Behavior of the node $S_j$**

```
Begin
  Broadcast (Candidature_Query, Neighbors)
  If Reception ($R_{Cn}$) Then
     Add ($R_{Cn}$, $T_{Cand}$)
  //Select the successor sink
  If Reception (all_rps) Then
     p = Compute $p^m(C_N = sink)$
     //p: Marginal distribution of probability
     $S_{j+1}$ = max(p)
    //Deduce the node within maximal probability
     Send ('You are the successor', $S_{j+1}$)
     Send (My_Pred' $S_{j-1}$', $S_{j+1}$) // If exists
End
```

The network resulting from the final description of the recovering path can be used to find out updated knowledge of the state of a subset of variables, when other variables (the evidence variables) are observed. Suppose we wish to compute the probability that there is an access to the base station for the specified node $S_1$, given that both its energy level and its neighbouring parameters are satisfied.

$$p(A_1|e_1, v_1) = \alpha\, P^J(A_1, e_1, v_1) =$$
$$\alpha \sum_{s_i,i=1}^{n} \sum_{e_j,j=2}^{n} \sum_{v_k,k=2}^{n} \sum_{a_l,l=2}^{n} P^J(s_i, e_j, v_k, a_l, a_1, e_1, v_1) \qquad (5)$$

Where $\alpha$ is a constant of normalization.

In eq.(5), we need to sum over all the values of the hidden variables. In general, this form of computation process is more complex. So, we need to introduce a variable elimination algorithm [16] that will help us avoid the duplicate computations. The first step is as follows:

$$p(A_1|e_1,v_1) = \alpha p(A_1)\sum_{s_1} p(s_1|A_1).p(e_1|s_1).p(v_1|s_1).\sum_{A_i,i=2,n} p(A_i).\sum_{s_j,j=2,n} p\left(s_j\middle|A_j,S_1\right).p\left(e_j\middle|s_j\right).p\left(v_j\middle|s_j\right) \quad (6)$$

This process is repeated until obtaining a set of simple factors (sums) easy for computing.
The complexity of variable elimination depends on the size of the largest factor that must be created during the computation. This in turn depends on the order in which the variables are eliminated.
This process is a technique for inserting fault tolerance into systems. It basically consists of storing the current application state, and later on, uses it for restarting the execution in case of failure. So, in this process the sink $S_j$ takes checkpoints regularly (i.e. the consistent state of the sink is saved). In addition of this state, the energy level will be sent to the successor sink $S_{j+1}$**.** This process happens in the following situations:

- **a)** After the step of aggregation is accomplished at level of the node $S_j$**.**
- **b)** When a request of checkpointing query sent by the sink $\mathbf{S_{j+1}}$ is received.
- **c)** At every elapsed time θ2, defined initially before the deployment step.

Indeed, the sink $\mathbf{S_{j+1}}$ receives the consistent state of $S_j$. This state is systematically saved in the table of checkpointing ($T_{CP}$) as a stable storage. A backup of the table provides the required recovery of $S_j$ failure. This process of checkpointing is well described in the Algorithm 4.

**Alg4: Behavior of $S_j$ within the Checkpoint/Restart Process**

```
Begin
  Send (Initialisation_Query)
  Repeat
    Initialise(θ1, θ2, Sensing_Data_Table )
    If Reception(Interest_Query, SB) OR Timeout (θ1) Then
        Broadcast ( Interest_Query)
    If Reception (Interest_Response, NSj)) Then
        Add ( Interest_Response ,Sensing_Data_Table)
      //Computing and agregation
        Aggregate (Sensing_Data_Table)
        Send(Aggregation_Result, SB)
        CP = Take_CPj() ; Send (CP, S_{j+1})
        E = Compute_Energy_Level(); Send (E , S_{j+1})
    If Reception (Request_Query_CP,Sj+1) OR Timeout (θ2)
    Then
        CP = Take_CPj() ; Send (CP, S_{j+1})
        E = Compute_Energy_Level() ;Send (E, S_{j+1})
End
```

## 3.4 Failure detection process

The failure detection of the node $S_j$ is ensured by the node $S_{j+1}$ which represents the successor node of the actual node $S_j$. Two techniques are proposed. The first one is a similar technique to the conventional ping. At every time $\theta_3$, the node $S_{j+1}$ tests periodically the activity of $S_j$, by sending messages as "Hello$_q$" form (q: number of sequence). If there is not any response after **β** attempts, then the node $S_j$ will fail. The second technique of detection is based principally on the energy consumption. In this case, the node $S_{j+1}$ evaluates the energy level (E) of $S_j$.
If $E \leq E_{threshold}$ then a failure will be detected at the level of the node $S_j$ .
The information concerning the failure of $S_j$ is distributed through the network to reduce the impact of the failure (i.e. for the future, no message will be sent to the failure node). So, we present these techniques in detail in the Algorithm5.

**Alg5: Behaviour of $S_{j+1}$ within failure detection /recovery Process**

```
Begin
  Repeat
    If timeout (θ4) Then Send (Request_Query_CP, Sj)
    If Reception (CP,Sj) Then Save(CP, T_CP)
  Until (end of queries)
  // Detection and failure recovery
  If timeout (θ3) Then Send ('Hello', Sj)
  If β attempts OR E < E_threshold Then
      Broadcast ('Sj is failed'),
      Extract_Last_CP (T_CP),
      Install(Last_CP),
      Distribute ('I'm the Sink').
End
```

## 5. Experimental Results

In this section, we describe the experiments that were performed to evaluate the effectiveness of our protocol with the Simulator PowerTOSSIM of TinyOS. It allows simulating the behavior of a sensor (sends / receives messages via radio waves, information processing ...) within a wireless sensor network.
PowerTOSSIM can be used with a graphical interface TinyViz, for viewing intuitively the behavior of each sensor network. The interesting feature is that it offers the possibility of knowing the energy consumption of each component of the sensor, such as the CPU, memory, LEDs and especially the Radio antenna.
During the simulation phase, we used the energy model of the sensor nodes Mica2 [17]. Several scenarios are executed by varying at each time the number of nodes, the running time, and the failures number in the network. This allows analyzing each parameter assessment to study the impact of the proposed solution over the lifetime of the network.

### 5.1 Initialization time of the network

We assume in the rest of this experimental section that the number of nodes deployed varies between 50 and 500 nodes. We present the results of two specific networks. The first one concerns our proposed approach (ASFT) which is tested at each sensor node. While the second concerns simply the OWSN ordinary wireless sensor network. The figure 6 shows that the initialization time increases as the size of the network increases. Time of initialization of the protocol ASFT is less than the one of OWSN. This means that the proposed solution affects slightly the initialization time of the different nodes.

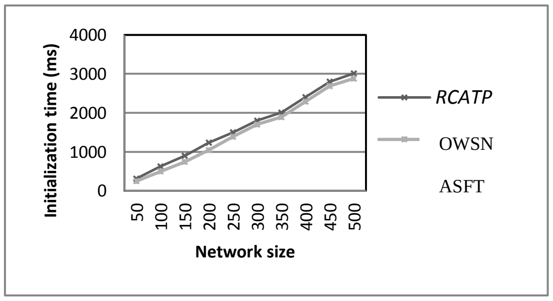

Fig.6 Time for initializing a WSN

### 5.2 Energy Consumption

The nodes of a WSN are powered primarily by batteries. They must work with a frugal energy outcome. In addition, they more often have a lifetime of the order of several months, see a few years, since the replacement of the batteries is not easy for networks with thousands of nodes. The consumed energy computed depends of the various components integrated in the nodes. The parameters of this simulation are: 500 seconds as the simulation time and 50 nodes deployed in the network. We show in Figure 7, that the node consuming more energy is the node $S_j$. This is due to the additional function of collecting and transmitting data to the base station. Therefore, we remark that the components consuming the most energy are the CPU and the radio. As the processing increases, the consumption increases. Indeed, the transmission and the reception of messages consume much more energy.

### 5.3 Recovery time & consumption energy

The main idea is to minimize the recovery time and consuming less of energy, once a failure of a sink Sj has been detected. It is therefore necessary to analyze these two parameters at the same time.

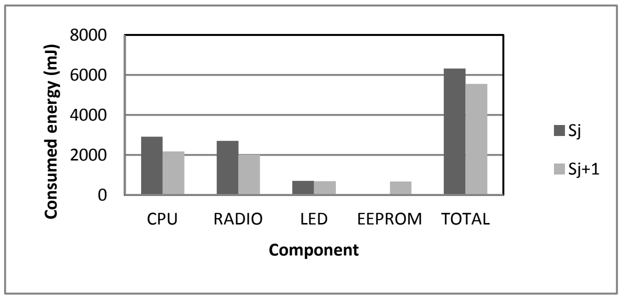

Fig.7 Energy consumption by different components of each node of the network

The recovery time is determined by the period between the occurrence of the failure and its recovery. During the simulation (at 250s of simulation time), we inject a failure in the sink $S_j$. Once this is done, we compute the consumed energy and the restart time. The Figure 8 shows that the recovery consumption energy of the node $S_{j+1}$ has increased after injecting the failure in $S_j$. This obtained event explains well the recovery process. We also note that the curve of the $S_j$ is stabilized (any information about $S_j$) that is due to the injected failure. So, we can see that the recovery time for this simulation is reduced. These results are very satisfying which implies an efficient recovery of the failed node $S_j$ by $S_{j+1}$ .

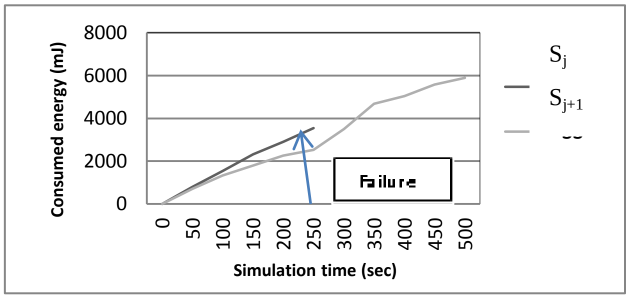

Fig.8 Consumption Energy of $S_j$ and $S_{j+1}$ (injected failure at 250s)

## 6. Conclusions and future work

In this paper, we focus on the fault tolerance problem in a wireless sensor network. For this purpose, we have developed a probabilistic checkpointing protocol. In this approach, we propose different process which composed our work. We used a causal network for modeling the behavior of the sink. Based on this model and the technique of checkpointing/restart that restores the consistent state saved earlier, we have created a recovering path for the failed sink. Indeed, the recovery process is triggered after failure's detection process. This protocol can improve the performance of the considered sensor network. Several aspects of the sensor network tolerance fault problem remain challenging, and worthy of continued study. Taking in consideration the mobility of the nodes in the wireless sensor network, we can model our problem through a dynamic Bayesian network and then applying the inference methods. Another interesting work is to implement the proposed protocol on a real system.

## References

[1] V. Shnayder, M. Hempstead, B. Chen, A. Geoff Werner, and M.Welsh, PowerTOSSIM: Simulating the Power Consumption of Large-Scale Sensor Network Applications: Div. of Eng. and Applied Sciences Harvard University, Nov. 3–5, 2004, Baltimore, Maryland, USA.

[2] Ian F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and Erdal Cayirci, A Survey on Sensor Networks, IEEE Communications Magazine. Pages 102-114, August 2002.

[3] P. Maurizio, L. Paladina. A Multi-Sink Swarm-based Routing Protocol for Wireless Sensor Networks, 978-1-4244 -4671-1/09, 2009 IEEE.

[4] A. Avizienis, J.C. Laprie, B. Randell et C. Landwehr, Basic Concepts and Taxonomy of Dependable and Secure computing, IEEE Transactions on Dependable and secure computing, 1 (1), pp.11-33, 2004.

[5] A Survey on Fault Tolerance in Wireless Sensor Networks, Luciana Moreira S. de Souza, SAP Research, Karlsruhe, Germany.

[6] S. Harte and A. Rahman, Fault Tolerance in Sensor Networks Using Self-Diagnosing Sensor Nodes. In The IEE International Workshop on Intelligent Enviroment, pages 7–12, June 2005.

[7] W. Xiaobing, Ch. Guihai. Dual-Sink: Using Mobile and Static Sinks for Lifetime Improvement in Wireless Sensor Networks. 1-4244-1251-X/07, 2007 IEEE. [8] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi, A Discrete-Time Battery Model for High-Level Power Estimation. In Proceeding of the Design, Automation and Test in Europe Conference and Exhibition 2000, pages 35–39, 2000.

[8] D. Rakhmatov and S. B. Vrudhula., Time-to-Failure Estimation for Batteries in Portable Electronic Systems. In Proceedings of the 2001 international symposium on Low power electronics and design, pages 88–91, 2001.

[9] L. B. Ruiz, I. G. Siqueira, L. B. Oliveira, H. C. Wong, J. M. S. Nogueira, and A. F. Loureiro. Fault Management in Event-driven Wireless Sensor Networks. In Proceed. of the 7th ACM intl. symp. on Modeling, analysis and simulation of wireless and mobile systems, pages 149–156, 2004

[10] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, Li Yin, and F. Yu, Data-Centric Storage in Sensornets with GHT, a Geographic Hash Table. *Mob. Netw. Appl.*, 8(4):427–442, 2003.

[11] C. Y. Wan, S. B. Eisenman, and A. T. Campbell, Coda: Congestion detection and avoidance in sensor networks. In Proceedings of SenSys, 2003.

[12] J. Considine, F. Li, G. Kollios, and J. Brers, Approximate aggregation techniques for sensor databases. In Proceedings of IEEE ICDE, 2004.

[13] A. Akbari, D. Arash, Detection and Recovery in Wireless Sensor Network Using Clustering. International Journal of Wireless & Mobile Networks (IJWMN) Vol. 3, No. 1, 2011.

[14] Pearl, J., Probabilistic Reasoning in Intelligent Systems, Morgan Kaufmann, San Francisco, CA, 1988

[15] Jensen, Finn V; Nielsen, Thomas D. Bayesian Networks and Decision Graphs. Information Science and Statistics series (2nd ed.). New York: Springer-Verlag.ISBN 978-0-387-68281-5., 2007

[16] S. Russel, P. Norvig, A modern Approach of artificial intelligence,

[17] CrossBow Products: http://www.xbow.com/products.