Energy-Efficient Fault-Tolerant Mechanism for Clustered Wireless Sensor Networks

Yongxuan Lai[1,2], Hong Chen[1,2]
[1]School of Information, Renmin University of China, Beijing, P.R.China
[2]Key Laboratory of Data Engineering and Knowledge Engineering, MOE
Email: laiyx@ruc.edu.cn, chong@ruc.edu.cn

***Abstract*—Clustering is an effective topology control and communication protocol in wireless sensor networks ("sensornets"). However, the harsh deployed environments, the serious resource limitation of nodes, and the unbalanced workload among nodes make the clustered sensornets vulnerable to communication faults and errors, which undermine the usability of the network. So mechanisms to improve the robustness and fault-tolerance are highly required in real applications of sensornets. In this paper, a distributed fault-tolerant mechanism called CMATO (Cluster-Member-based fAult-TOlerant mechanism) for sensornets is proposed. It views the cluster as an individual whole and utilizes the monitoring of each other within the cluster to detect and recover from the faults in a quick and energy-efficient way. CMATO only needs the local knowledge of the network, relaxing the pre-deployment of the cluster heads and a *k*-dominating set (k>1) coverage assumptions. This advantage makes our mechanism flexible to be incorporated into various existing clustering schemes in sensornets. Furthermore, CMATO is able to deal with failures of multiple cluster heads, so it effectively recovers the nodes from the failures of multiple cluster heads and the failures of links within the cluster, gaining a much more robust and fault-tolerant sensornets. The simulation results show that our mechanism outperforms the existing cluster-head based fault-tolerant mechanism in both fault coverage and energy consumption.**

Keywords - fault-tolerant mechanism; energy-efficient; clustering; wireless sensor networks

I. INTRODUCTION

In wireless sensor networks ("sensornets") [1], it is well known that clustering is an effective and useful mechanism for topology control and data gathering [2], [3], [4]. In clustering scenario, the network is partitioned into clusters, and cluster heads that act as a "backbone" of the network are selected. Each node in the "backbone" merges data from its member nodes, so it gains much reduction of data transmissions. Many applications on sensornets, such as data gathering, data aggregation, are based on clustering.

However, clustering also brings about some problems for the sensornets, especially when considering the fault-tolerance and robustness issues in the sensornets. Many sensornets are often deployed in harsh and poor environments, such as the battlefield, forests, etc. The changing external environment and the highly resource-constrained nature of nodes make the network vulnerable to various interference. For example, the weather sensitive nodes may fail and stop working in rainy days, making it hard to maintain the clustered structure of the network. Meanwhile, the cluster heads, who assume more tasks in clustered sensornets, have a larger probability of errors. Once cluster head is failed, it would prevent transmitting data outside in a cluster or even larger scale. As Fig.1 shows, once the cluster head *E* fails, the whole cluster would be in a failure, leading to the incapability of its members transmitting data outside. Even worse, since packets generated by cluster *F*, *H*, and *I* are relayed through *E*, these clusters would also be in a state of failure unless a fault-tolerant mechanism is used. As faults are inevitably exists in sensornets, fault-tolerant mechanisms are crucial and highly required for the applications in the network.

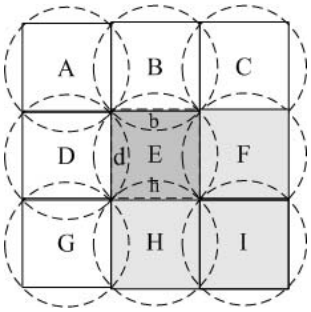


Fig. 1. Illustration of faults in clustered sensornet . Rectangles of *A*, ..., *I* denote the clusters, the circles represent the communication ranges of the cluster heads. Dark areas are affected by the failure of the cluster head of *E*.

Some protocols have been proposed to combat the faults in the clustered sensornets either through the pre-computation of cluster head placements or the cluster-head based run-time monitoring. However, both of these schemes have their drawbacks. The *k*-fold minimum dominating set (*k*-MDS) scheme, which belongs to the scheme of pre-computation of cluster head placements, needs to know the global knowledge and pre-calculate the locations of the cluster head centrally; while for the cluster head monitoring scheme, it can only detect and recover from a single cluster head failure. We argue that the overhearing techniques could be utilized in the fault-tolerant mechanisms in sensornets. Due to the broadcasting nature of wireless transmissions, many nodes in the vicinity of a sender node overhear its packet transmissions even if they are not the intended recipients. Several energy efficient MAC

This work is supported by the National Natural Science Foundation of China under Grant No.60673138.

1-4244-1251-X/07/$25.00 2007 IEEE.

protocols, such as WiseMAC [5], Wake-Up-Frame [6], and SyncWUF [7], have been proposed in MANET (mobile ad hoc network) or sensornets. They switch the RF (radio frequency) transceiver, which is one of the biggest power consumers in a sensor node, to low-power sleep mode as much as possible, yet they can wake up for a very short time randomly or periodically, saving a factor of tens or even hundreds of energy. So it is cheap for the nodes to aware the liveness of other nodes and run the fault-tolerant mechanisms through these new kind of overhearing techniques in sensornets.

In this paper, we present a distributed fault-tolerant mechanism called CMATO (Cluster Member based fAult-TOlerant mechanism) in sensornets. In CMATO , the nodes within the cluster are all involved in the processing, they monitor the links to the cluster head and overhear the transmissions of the cluster heads who are in neighborhoods. When a cluster head fails, its cluster members will detect it and re-select a new cluster head for the cluster; when the link from the cluster head to the cluster member is broken, the member would transmit itself to the neighboring cluster for recovery. Both the detection and recovery are operated locally in the running time. Simulation results show that CMATO can be embedded into the existing clustering algorithms to detect the faults in the nodes, and then dynamically recover the network from faults in an energy-efficient way.

The following sections are organized as follows: section II summarizes the related work. Section III gives some definitions and the models discussed in this paper. Section IV presents the detailed fault-tolerant mechanism. Section V describes the experimental setup and evaluates the performance of CMATO . Finally, section VI concludes the paper.

II. Related Work

Several clustering algorithms have been proposed in sensornets. In LEACH [2], each node has an equal chance to become a cluster head in each round. The cluster forming process is to finish within K (a constant) steps, and then each cluster head collects data from its members and communicates directly with the base station. An improved version to LEACH, LEACH-C [8] uses a centralized clustering algorithm to produce better clusters, thus achieves better performance. TPC [3] (Two-Phase Clustering scheme) contains two phrases in clustering process. In the first phrase of TPC, it uses a LEACH-like clustering algorithm to generate cluster heads. In the second phrase, also called cluster restructuring, each cluster member searches for a neighboring node within the cluster to set up an energy-saving relay link. Then data are aggregated on this relay link to reduce the intra-cluster communication cost. HEED [4] extends LEACH by incorporating communication range limits and intra-cluster communication cost information. The probability for each node to become a tentative cluster head depends on its residual energy, and final cluster heads are selected from the tentative cluster heads according to the intra-communication cost. All these clustering algorithms focus on the balanced energy consumption mechanism and efficient cluster forming algorithms for the sensornets. Unfortunately, they pay little attention on the fault-tolerant issues in the clustered network.

For the fault-tolerant mechanism, [9] proposes a multi-path routing in sensornets. It builds multiple routing paths from the source to the destination, so transmissions can switch to another path when a path is failed. [10] investigates the problem of cluster head placements so that every node can communicate with at least two neighboring cluster heads. When a cluster head is failed, the node can transfer itself to the other cluster head for transmissions. In [11], it converts the node placement problem into the *k* minimum dominating set (*k*-MDS) problem. The goal of *k*-MDS is to find the minimal subset of the nodes as cluster heads so that every node can communicate with at least *k* cluster heads. When *k*=1, it is the minimal coverage problem; when *k*=2, it is the problem discussed in [10]. Unfortunately, the base station needs to know the global knowledge and pre-calculate the location scheme by centralized processing. In [12], Gupta etc. proposed a fault-tolerant mechanism based on inter-cluster monitoring. When a gateway, say *I* in Fig.1, can not communicate with some gateway *E*, *I* would consult to its neighboring gateways *F* and *H* whether *E* is failed. If both *F* and *H* confirm its broken link with *E*, gateway *I* would declare the failure of *E*. Then the neighboring cluster heads would further negotiate with each other to transfer the nodes of *E* into their clusters. The main drawback of this approach is that it can not deal with the failures of multiple gateways because of the mutual inter-cluster-head consulting mechanism. Moreover, the nodes would have to transmit the control messages in a large distance when running the cluster-head based mechanism.

III. Preliminaries

A. Network Model

Let us consider a sensor network which consists of N nodes uniformly deployed over a square area. We denote the i^{th} sensor by n_i and the whole node set S = $\{n_1, n_2, .., n_N\}$, where $|S| = N$. There is a base station (i.e., sink node) located in the field, and the nodes use multi-hop routing to send data to it. We assume all nodes, including the cluster heads and the normal nodes, are homogeneous and have the same capabilities, and they use power control to vary the amount of transmission power which depends on the distance to the receiver. We use $d(n_i, n_j)$ to denote the distance from node n_i to node n_j, and bi-directional symmetrical links are assumed. We then provide three definitions for the convenience of discussions in the following sections:

Definition1 (Neighbor). $Nbr(n_i, r)$ is the *r*-range *neighbor* set of n_i, if $\forall n_j \in Nbr(n_i, r)$, s.t. $d(n_i, n_j) \leq r$. n_i and n_j are *r*-range neighbors of each other.

Definition2 (In-cluster Neighbors). $INbr(n_i, r)$ is the *r*-range *in-cluster neighbor* set of n_i, if $\forall n_j \in INbr(n_i, r)$, s.t. $i \neq j$, $n_j \in Nbr(n_i, r)$ and $CH(n_i) = CH(n_j)$, where $CH(n_i)$ denotes the cluster of node (n_i). n_i and n_j are *r*-range in-cluster neighbors of each other. Of course, the cluster head is always the in-cluster neighbor of its members.

Definition3 (Neighboring Cluster Head). $NCH(n_i, r)$ is the

r-range *neighboring cluster head* set of n_i, if $\forall ch_j \in NCH(n_i, r)$, s.t. $ch_j \in CHS \bigcap Nbr(n_i, r)$ and $ch_j \neq CH(n_i)$, where CHS denotes the set of all cluster heads and $CH(n_i)$ denotes the cluster head of n_i. ch_j is the neighboring cluster head of n_i.

For example, in Fig.2(a), n_1, n_4, n_5 are in-cluster neighbors of n_6, because they are neighbors of n_6 and in the same cluster with n_6. In Fig.2(b), the node *m* has a neighboring cluster head $B.ch$ because *m* and $B.ch$ are neighbors of each other.

B. Fault Model

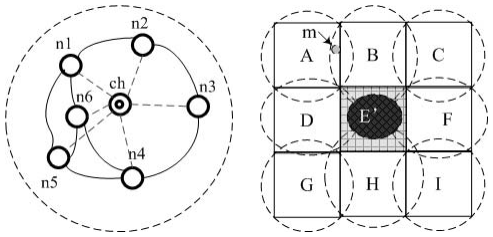
Due to the highly limited resource of the nodes and harsh deployed environments, sensornets are vulnerable to various communication faults and errors. These are different ways to classify these faults into different categories based on different criteria. According to the time the faults last, the faults could be divided into ephemeral, intermittent, and permanent faults. *Ephemeral faults* relate to faults who happen in a very short time and the nodes can recover automatically by themselves, such as the temporal disconnection of nodes when a vehicle passes by. *Intermittent faults* are similar to ephemeral faults, only they last for a longer time before the nodes automatically recover from these faults. The *permanent faults* are the most serious ones. Damages and crashes of nodes caused by men or natural disasters belong to this kind. Once they happen, the nodes can not revert to the original state, cutting down the connections forever. According to the structure of the communication, we could divide the faults into *medium faults* and *nodes faults*. The medium faults happen when there are some interference in the medium which the wireless link depends on. For example the link may be cut off when there are barriers just located in the path between two nodes. The node faults are the failure or errors of the nodes themselves, caused by the hardware or software mistakes or external damages. In clustering scenario, when the nodes who serve as cluster heads are failed, the whole cluster are disabled to communicate outside, highly reduce the availability of the sensornets.

Note that intermittent faults can also viewed as permanent faults because during the time the faults happen, the network are in failure and are need to be recovered so as the data could be transmitted outside. Due to the large impact of the permanent (and intermittent) faults in the cluster head side, in this paper we explore the fault-tolerant mechanism for the permanent faults in the cluster head side, as well as the medium faults between the cluster head and cluster members.

IV. Framework

A. Overview

We could see how our fault-tolerant mechanism CMATO works in Fig.2. In Fig.2(a), the nodes within a cluster overhear their cluster head ch and their in-cluster neighbors. So the cluster members could detect the failure of the cluster head quickly. In Fig.2(b), when all the nodes aware the failure of the cluster head of *E*, some nodes (nodes located in the light gray area) join the neighboring clusters, while others



(a) Detection (b) Recovery

Fig. 2. Overview of the CMATO framework. (a) Cluster members $(n_1, n_2, ..., n_6)$ are overhearing the cluster head ch and its in-cluster neighbors. (b) When the cluster head of *E* is failed, nodes in the light gray areas join the neighboring clusters , and nodes in the in dark gray areas join the newly constructed cluster *E'*.

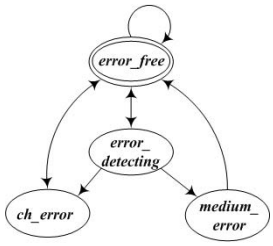


Fig. 3. State transition graph. *error_free* is the normal state of the nodes, *error_detecting* is the fault detection state, *ch_error* and *medium_error* are the result state of the detection, indicating the types of faults.

(nodes located in dark gray area) join the newly constructed cluster E' . Moreover, there are 4 states for the nodes: *error_free*, *error_detecting*, *ch_error*, and *medium_error* (see Fig.3). The transition from *error_detecting* to *error_free*, *ch_error*, or *medium_error* is the error detection phrase, while from *ch_error* or *medium_error* to *error_free* is the fault recovery phrase. We will discuss each of them in the following subsections.

B. Fault Detection

Fault Detection is the first phrase in our fault-tolerant mechanism. We consider the cluster as a whole, which could detect the faults of its own. There are many cases in which the cluster members could aware the aliveness of their cluster heads. For example, when the cluster members send the sensed data to the cluster heads, the cluster heads may reply through short acknowledged packets. Or the nodes would actively broadcast beacon messages for synchronization, and these messages could be used to indicate the aliveness of the cluster heads. Especially, in the Wake-up-Frame MAC scheme [6] when the cluster is relaying packets for its neighboring cluster heads, the cluster head would send lots of very short wake-up frames, which could be cheaply overheard by the cluster members through periodical but very short wake-ups.

In our CMATO mechanism, a node is overhearing the transmissions of its cluster head and its in-cluster neighbors when in the *error_free* state. If it can not overhear any short frames or short packets indicating the aliveness of its cluster

head, it would be waken up to transfer to the *error_detecting* state which keeps its receiver open. Then this node has to decide whether the failure of the cluster head, or the fault in the medium has caused the the broken link to the cluster head.

In CMATO , we use an *in-cluster consulting mechanism* to correctly classify the type of the faults. The principle of the CMATO in-cluster consulting mechanism is: if more than α (a predefined threshold) percents of the nodes in the cluster detect the failures of the links to the cluster head, then the cluster head should be declared as a failure. Even if the cluster head is not failed but most of the cluster members can not communicate with their cluster head because of the medium faults, then we could also classify this case into the fault of the cluster head failure. Because in this case it is wiser for the whole cluster to re-select a new cluster head. CMATO uses *unable_list* to represent the list of cluster members who disconnect from the cluster head. Then *unable_list* is propagated and updated among the cluster members. If the length of the list is more than $\alpha \times |C|$ ($|C|$ is the number of member nodes in cluster *C*), then the cluster head is to be declared as failed.

We could see the detailed detection flow through an example in Fig.2(a). Assume the cluster head ch encounters a permanent fault at some time, and the whole cluster members could not overhear the short *beacons* of the cluster head, so every member transits its state from *error_free* to *error_detecting*. The cluster member adds its *id* to its *unable_list* ($unable\_list(i) = \{i\}$), sets a random timeout timer (t_{id}) for broadcasting the *unable_list*, and keeps its receiver open. Suppose the order for the timers is $\{t_3, t_1, t_6, t_4, t_2, t_5\}$. Then at t_3, the timer is fired and n_3 broadcasts a *ch-unconnect* message which wraps the $unable\_list(3) = \{3\}$. The in-cluster neighbors of n_3, which are n_2 and n_4, receive this message and extract the list for a union of its own *unable_list*. Then $unable\_list(2) = \{2, 3\}$, $unable\_list(4) = \{4, 3\}$. All the sender and the receivers of that message reset their timers: the sender creates a new timer, while nodes who increase their $unable\_list$ length set the times of their timers shorter. Suppose the new order of the timers is: $\{t_1, t_4, t_6, t_2, t_5, t_3\}$, then n_1 broadcasts a *ch-unconnect* message that wraps $unable\_list(1) = \{1\}$ at t_1. In the similar way, the timers and the $unable_list$s of in-cluster neighbors are reset. That is: $unable\_list(2) = \{2, 3, 1\}$, $unable\_list(6) = \{6, 1\}$, and $unable\_list(5) = \{5, 1\}$. Suppose the new order of the timers is: $\{t_4, t_6, t_5, t_2, t_3, t_1\}$, then at t_4, after the broadcasting of *ch_unconnect* message that wraps $unable\_list(4) = \{4, 3\}$, the $unable_list$s are reset as: $unable\_list(3) = \{4, 3\}$, $unable\_list(6) = \{6, 1, 4, 3\}$, and $unable\_list(5) = \{5, 1, 4, 3\}$. Because the $unable\_list(6).length = unable\_list(5).length = 4 > 6 \times 60\%$, then n_6 and n_5 would declare the failure of the cluster head by broadcasting a *ch-fail* message. The nodes who receive this message for the first time would relay it immediately so that all the cluster members would know that their cluster head is failed. Cluster members who receive the *ch-fail* message would enter the recovery phrase immediately.

However, if ch is not failed, yet n_4 detects that the link to the cluster head is broken, and enters into the *error_detecting* state and broadcasts the $unable\_list(4)$ at some time. In this case, since the in-cluster neighbors of n_4 can still communicate with its cluster head, they would ignore the $ch - unconnect$ message from n_4. Then after some time interval (the maximal detecting time interval), since no *ch-fail* messages are broadcasted, n_4 would figure out that the disconnected link to its cluster head is due to the faults in the medium of the link. So it transits to the $medium\_error$ state.

C. Recovery from Faults

For the recovery from the faults, every node maintains a neighboring cluster head set *NCH* (see definition 3). As mentioned in subsection B, in CMATO when the cluster heads are communicating with each other or periodically broadcasting beacon messages, the cluster members who overhear their transmissions or beacon messages would aware the aliveness of these cluster heads. So the cluster members can simply add the neighboring cluster heads into their *NCH*s. For example, in Fig.1, the nodes that locate at region b, f, h, d in cluster *E* would add the cluster head of B, F, H, D into their NCHs respectively. When the cluster members in regions $\{b, f, h, d\}$ find the cluster head of *E* is failed or the link to the cluster head is broken, they could transfer themselves to their neighboring clusters. However, the *NCH* of the nodes located in region $\{E - (b + f + h + d)\}$ would be empty, so new cluster heads need to be selected for the recovery of these nodes.

1) Recover from the medium_error: We first discuss the recovery from medium faults since it is the basis for the cluster head failures. If the cluster member figures out it is a medium error, it would try to join into its neighboring clusters. If the neighboring cluster head set is not empty ($|NCH| \geq 1$), it just sends a *join-request* message to the "best" neighboring cluster head, where different criteria could be used to weigh among the neighboring cluster heads. On receiving this message, the neighboring cluster head would acknowledge this message and prepare for receiving data from this node, so the node is recovered. If the neighboring cluster head set is empty ($|NCH| = 0$), then the cluster member has to select a node from its neighbors as the *relay* node, through which the cluster member would send data to the cluster head. We argue that the recovery through neighboring cluster head is better than through the relay node. Because waking up the node to become a relay node is actually making that node to be a cluster head. This method would increase the number of the cluster heads and increase the interference among clusters when multiple medium faults happen in the network.

In the cluster head side, it would assume the cluster member whose link to the cluster head is disconnected for a predefined time to be failed. So it just removes that node from its cluster member list.

2) Recover from the ch_error: If cluster head is declared failed, all the cluster members would be notified through the broadcasted *ch-fail* messages. Then the cluster member would

compete to be a cluster head through a weight function f:

$$f(c_{im})_r = r \times \frac{|INbr|}{|C_i|} + (1-r) \times \frac{E_{cur}(c_{im})}{E_{max}(C_i)}$$

where c_{im} is the m^{th} cluster member of cluster C_i, $|INbr|$ is the size of the in-cluster neighbor set (see definition 3), $|C_i|$ is the size of the cluster, so $\frac{|INbr|}{|C_i|}$ represents the *connectivity* factor. $E_{cur}(c_{im})$ is the residual energy of that node, and $E_{max}(C_i)$ is an estimate of the maximal residual energy among the nodes in the cluster C_i, so $\frac{E_{cur}(c_{im})}{E_{max}(C_i)}$ represents the *energy* factor. Note that $E_{max}(C_i)$ could be estimated in the cluster formation phrase, since the information of residual energy are to be exchanged in some clustering protocols. r (0.6 in this paper) is a predefined bias parameter that combines the connectivity and energy factor into the final weight of becoming a cluster head. The larger of the weight, the more probability for the node to become a new cluster head.

According to the weight function f, when a node decide to be a cluster head, it would broadcast a cluster head advertisement message $ch-adv$ just as the cluster formation phrases do in some clustering algorithms, such as TPC [3]. Note that timer based cluster head selection is applicable here. The larger the weight, the shorter time that node would broadcast a $ch-adv$ message to notify its neighbors that it is the cluster head. On receiving this $ch-adv$ message, the nodes would cancel their timers of becoming cluster heads, and insert the id of the sender into their NCHs. When new cluster head is selected, the members in the failed cluster would update their NCHs and send the $join-request$ messages to the "best" cluster heads in the NCHs, or to the neighboring relay nodes if the NCH is empty. To avoid too many collisions of these request messages, each node would set a timeout based on its id to defer the sending of the messages. When a collision happen, the node would reset its timer for a random delay.

When the neighboring cluster head receives the request message, it just accepts the request and adds the node into its member list. So some of the nodes would join the newly selected cluster head, while others would join the neighboring cluster heads. In extreme cases, there would be nodes whose NCH is empty even after the new cluster head has been selected. In these cases, these nodes have to rely on the relay nodes for the recovery, which is discussed in subsection C.1.

V. Performance Evaluation

A. Environment Setup

To validate the effectiveness of our fault-tolerant mechanism, we implement CMATO in the J-Sim [13] platform with wireless sensor network extension. In these experiments, there are 100 nodes randomly deployed in a rectangle field 200×200 m^2, where the sink node is located at (0, 0). The network is clustered using the LEACH [2] and HEED [4] clustering algorithms, the cluster heads then organize into a spanning tree for routing. The cluster head selection rate is 0.2, and cluster range is 50m. The links are bi-directional, and every node can communicate with the nodes within its communication range.

TABLE I
Parameters and their values

Parameter	Value
Electronics energy (E_{elec})	50 nJ/bit
Threshold distance (d_0)	87 m
Transmit Amplifier ($\epsilon_{fs}, \epsilon_{mp}$)	$10.00pJ/bit/m^2$, $0.0013pJ/bit/m^4$
Data packet size	4000 bytes
Beacon packet size	20 bytes
Initial energy	1.0 J/battery

Then we design a fault generating scheme to test the robustness and behavior of the sensor network. Faults are injected into the network. For every cluster head, if the time arrival of the k^{th} fault is T_k, then the inter-arrival times are defined as follows:

$$X_1 = T_1, X_k = T_k - T_{k-1}, for\ k = 2, 3, ...$$

We suppose X_i is independent, identically distributed random variable, and belongs to the classical exponential distribution with rate parameter λ:

$$f_{T_i}(t) = \lambda e^{-\lambda t}, t > 0$$

According to [14], the inter-arrival time stream $\{X_1, X_2, X_3, ...\}$ actually form a Poisson process. λ equals $E(X_i)$, which is the expected value of X_i. Apparently, the larger the λ, the less frequent the faults happen. So for the cluster member, since its workload is smaller than that of the cluster head, we set its λ 10 times equal to the λ of the cluster head. When a cluster-head fault arrives, we assume it is the perpetual failure and the cluster head cut off any of its links to other nodes; when a medium error arrives at a cluster member, the link from the cluster member to the cluster head is cut off. All failed nodes are awaken and all broken links are reconnected after the faults last for $0.2 \times rts$, where rts denotes the rotation time span[1]. We also assume that the controlling messages in our fault-tolerant mechanism are error-free and correctly received by their destinations.

B. Simulation Results

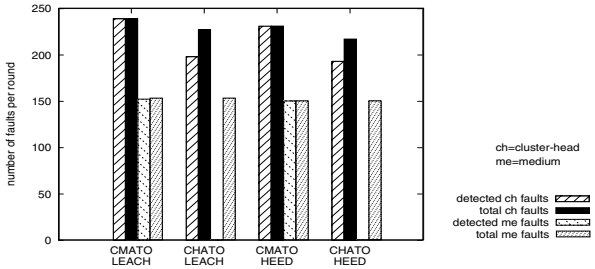


Fig. 4. Faults detected vs. total faults. ($\lambda = 800, rts = 2000$)

We implement CMATO in both LEACH and HEED protocol with the fault generator. Fig.4 compares the performance

[1]By doing so, CMATO avoids too many unconnected links and dead nodes in the running time.

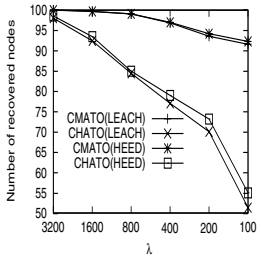


Fig. 5. Impact of λ to the number of nodes recovered from cluster head failures. $rts = 2000$

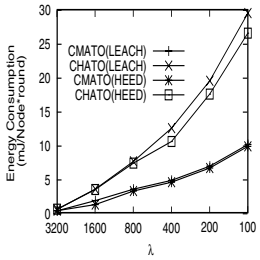


Fig. 6. Impact of λ to the energy consumption. $rts = 2000$

of CMATO with the CHATO (Cluster-Head-based fAult-TOlerant mechanism) when $\lambda = 800$ and $rts = 2000s$. CMATO can detect and recover from all the cluster head failures, while only about 87% of the faults are detected in CHATO . Furthermore, CMATO is able to recover from all the the medium faults, while CHATO provides no according mechanisms for the recovery from the medium faults. So in Fig.5 we only inject the faults of cluster head failures to further compare their performance. We vary the Poisson rate λ to compare the the number of recovered nodes, including cluster heads and cluster members in the failed cluster. When the faults are rare ($\lambda \geq 3200$), the network could be recovered by both the methods. However, when faults happen more frequently (smaller λ), more and more cluster heads are not recovered from faults in CHATO mechanism. This is because the CHATO can only deal with single fault, when multiple faults happen at the same time and probably in the neighboring clusters, CHATO would not work. More frequent cluster head faults also affects CMATO , but all these unrecovered nodes are cluster members whose neighboring cluster heads are dead, and they could not find a relay node.

We also study the energy consumption of our mechanism in our experiments with a simple energy dissipation model [8], the parameters of which are summarized at table I. We calculate the energy consumed for running the fault-tolerant mechanism apart from the other activities such as sensing and routing. The cost of the fault-tolerant mechanisms includes overhearing and exchange of the controlling messages. In Fig.6, we could see that the energy cost of CMATO is almost linear with λ. This is because the cluster could be viewed as an whole and deals with the fault individually, so the cost for fault detection and recovery is proportional to the number of faults (and the Poisson rate λ). While for the CHATO mechanism, when more faults happen and more clusters are failed, no new cluster heads are created. So the cluster-head based fault-tolerant mechanism has to enlarge its communication range for link monitoring and inter-cluster communications. So the line CHATO has a deeper slope than that of CMATO . The energy consumption of CMATO is more than 60 percent less than that of CHATO when $\lambda \leq 400$.

VI. Conclusion

In this paper, we have presented a distributed fault-tolerant mechanism called CMATO to improve the robustness of the clustered sensornets. CMATO views the cluster as a whole and takes advantage of the inter-cluster monitoring of nodes to detect the faults. When cluster members detect that the fault is a perpetual failure of the cluster head, they act cooperatively to select new cluster head to replace the failed one; when the cluster members detect the fault is a medium error, they just transfer themselves to the neighboring clusters or relay nodes. So it is a distributed algorithm that does not need the k-dominated set coverage assumption or any global knowledge of the network. Simulation results show that CMATO is effective to recover the sensornets from more than one cluster-head-failed faults at the running time, and is more energy-efficient than the cluster head based fault-tolerant mechanism.

References

[1] D. CRULLER, D. ESTRIN, and M. SRIVASTAVA, "Overview of sensor networks," *Computer(Long Beach, CA)*, vol. 37, no. 8, pp. 41–49, 2004.

[2] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, p. 10, 2000.

[3] W. Choi, P. Shah, and S. Das, "A framework for energy-saving data gathering using two-phase clustering in wireless sensor networks," *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pp. 203–212, 2004.

[4] O. Younis and S. Fahmy, "HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, 2004.

[5] A. El-Hoiydi and J. Decotignie, "WiseMAC: An Ultra Low Power MAC Protocol for Multi-hop Wireless Sensor Networks," *First International Workshop on Algorithmic Aspects of Wireless Sensor Networks*, 2004.

[6] X. Shi, G. Stromberg, Y. Gsottberger, T. Sturm, I. AG, and G. Munich, "Wake-up-frame scheme for ultra low power wireless transceivers," *Global Telecommunications Conference. GLOBECOM'04. IEEE*, vol. 6.

[7] X. Shi and G. Stromberg, "SyncWUF: An Ultra Low-Power MAC Protocol for Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 1, pp. 115–125, Jan.

[8] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[9] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-resilient, energy-efficient multipath routing in wireless sensor networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 4, pp. 11–25, 2001.

[10] B. Hao, H. Tang, and G. Xue, "Fault-tolerant relay node placement in wireless sensor networks: formulation and approximation," *Workshop on High Performance Switching and Routing (HPSR)*, pp. 246–250, 2004.

[11] F. Kuhn, T. Moscibroda, and R. Wattenhofer, "Fault-Tolerant Clustering in Ad Hoc and Sensor Networks," *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, 2006.

[12] G. Gupta and M. Younis, "Fault-tolerant clustering of wireless sensor networks," *Wireless Communications and Networking (WCNC 2003)*, vol. 3, 2003.

[13] A. Sobeih, W. Chen, J. Hou, L. Kung, N. Li, H. Lim, H. Tyan, and H. Zhang, "J-Sim: A Simulation and Emulation Environment for Wireless Sensor Networks," *IEEE Wireless Communications Magazine*, 2005.

[14] J. Sheng, S. Xie, and C. Pan, *Theory of Probability and Quantitive Statistics*, 2nd ed. Beijing, China: higher education press (China), 1989.