A Mixed-Method Approach to Fault Tolerance and Management for Resilient Hierarchical Routing

Ahmed Ismail, Karim Seddik
Electronics and Communications Engineering Department
American University in Cairo
Cairo, Egypt
email: a.m.ismail@aucegypt.edu, kseddik@aucegypt.edu

***Abstract*—The reliable transmission of information is an inherently difficult and unpredictable task for LEACH-based networks. This is due to their adoption of centralized data exchange points which are expected to deliver data transmission reliability without being compromised by increasingly high rates of energy consumption. Given that this is a goal made more difficult by the constraints intrinsic to WSNs, the hardware deployed, task to be performed and targeted environment, LEACH becomes an increasingly restrictive factor when designing for dependable data delivery. The limited number of fault tolerance measures available to LEACH creates the need for strong and tailored energy efficient fault tolerance methods. This paper therefore proposes a series of simple and low cost functions by which fault detection, tolerance and management may be achieved within a highly constrained implementation of LEACH. The protocol is then tested using a hardware deployment of 49 low cost nodes in order to verify the delivered benefits and costs.**

***Index Terms*—CH backup, duplicate nodes, fault tolerance, hardware, heartbeats, LEACH, link failure, WSNs.**

I. Introduction

Hierarchical routing has been a favourite for deployments given its long and proven history as a dependable class of classical routing protocols for WSNs. Out of these many protocols, LEACH, being the first energy-efficient hierarchical protocol, has received much focus by the research community. Although this attention has allowed for the materialization of dozens of LEACH-based routing techniques, the LEACH protocol in itself suffers from many limitations in the areas of fault detection, tolerance and management [1].

LEACH dictates that if a network starts out with nodes of equal energy levels, then nodes may self-elect themselves as cluster heads (CHs) based on the total number of nodes and the expected number of CHs. Otherwise, nodes require information on their own energy levels and the network's total energy as well. Once a node self-assigns itself the role of CH, the node advertises this new assignment by broadcasting it. Non-CH nodes receiving these broadcasts select the advertisement with the strongest RSSI and transmit a join request to its source. The CH creates a TDMA schedule from these join requests and broadcasts it such that nodes may know exactly when they may transmit. Once a round of transmissions is complete, the clusters dissolve and the process repeats.

The first problem with this procedure is that LEACH does not imply what a cluster ought to do once its CH fails. Left to its own devices, the cluster would remain isolated till the end of the round, at which point the nodes are able to once again reorganize themselves into functional clusters [2]. This implies that LEACH-based deployments are obligated to accept the inherent possibility that portions of its networks are to occasionally suffer from temporary periods of isolation.

The second identifiable problem is that non-CH nodes base their decision as to which CH to join solely off of advertisement messages' RSSI levels. This kind of behaviour does not take into account that the CH with the strongest RSSI may also suffer from time-dependent variations in its link quality. This means that LEACH may in fact assign the critical role of CH to nodes that are unable to provide consistently high packet delivery success rates.

With these problems in mind, recent literature has suggested a variety of fault tolerance methods which may be categorized as either centralized or distributed approaches. Centralized approaches include methods such as Sympathy, eScan, BOSS, and MOTE-VIEW [3], [4], [5], [6]. They provide accurate fault identification, but at the cost of using large volumes of messages. The algorithms employed by these techniques are also completely dependent on the sink node. If the sink node fails, or the network is partitioned, then the fault management process fails for either the entire network, or a part of the network, respectively. This has caused centralized approaches to be identified as risky and high-cost techniques [7].

Alternatively, distributed approaches try to overcome the limitations of the centralized methods by assigning the responsibility of the fault-related decision-making process to the entire network [8]. This is done by requiring nodes to perform self-detection in addition to neighbor monitoring and coordination [8]. Methods such as AppSleep [9], DSN [10], and the energy level management algorithms of [11] and [12], are based off of complex functions which require extensive memory usage, driving up the nodes' power consumption rates [7].

Less processing-intensive distributed approaches, as suggested by recent literature, have focused mainly on the use of three strategies; heartbeat messages, CH backups, and energy-level self-monitoring [13], [14], [15], [16], [17].

The first of these methods requires that each CH transmits periodic heartbeat messages to an assigned backup [13]. If the backup does not receive a heartbeat from its CH, the backup assumes that the CH has failed and attempts to replace it [13]. The first problem with this strategy is that it does not outline how a backup is selected for a CH, nor does it define how the cluster is meant to behave if the backup fails before the CH. Another flaw in this approach is concerned with the significant overhead associated with the use of continuous heartbeat messages. Also, another limitation of this algorithm is that if the keep alives fail for any reason other than the actual failure of the CH, then the backup CH will unnecessarily rise up to fill the already occupied network role of CH. Having two nodes occupy the same CH means that every message destined to be relayed by a CH would then be relayed twice wasting energy in the already resource constrained WSN.

The second strategy requires that each CH monitors its own energy level. Once the CH's energy level drops below a certain threshold, the node is meant to report its imminent failure to the backup so that the backup would immediately replace the failing CH without any incurred interruptions to the cluster's operations [13]. However, this method assumes that the CH will have enough resources, by way of channel availability and remaining energy, in order to be able to report that it is failing. Having said that, the findings of [18] show that the voltage of a discharging battery remains fairly constant throughout the majority of its lifetime before dropping sharply towards the end of its lifespan. The results of [18] therefore support the notion that nodes will most probably not have enough time or resources to inform its backup that it is failing before ceasing to function.

In order to overcome the limitations associated with the aforementioned fault detection and fault tolerance methods, this paper modifies the concepts of heartbeat messages and CH backups in order to detect CH failures and subpar node-to-CH connections. These methods are tested using 49 low cost nodes pre-loaded with a strictly constrained LEACH-implementation. This is in order to evaluate the fault tolerance methods' contributions to the network's lifetime, robustness, data delivery success rates and operating overhead.

Consequently, Section II of this paper details the developed fault tolerance methods in conjunction with a description of the hardware and routing layer used to verify the performance of these methods. Section III details the tests performed and provides an evaluation of the algorithms' capabilities based on the results attained. Finally, Section IV concludes the paper by summarizing the lessons learnt and the recommended conditions under which the proposed methods may be deployed.

II. System Design

This section is mainly concerned with describing the full system used to prototype and test the proposed algorithms. The first subsection covers the constrained LEACH implementation used for prototyping and testing. The second subsection gives an overview of the various fault tolerance functions introduced to the LEACH protocol. The third subsection describes the hardware used to form the WSN nodes and the testing bed.

A. Constrained LEACH-Based Routing Layer

In order to test the performance of the proposed algorithms within harsh routing conditions, the designed routing layer has been constrained in several ways. Although these constraints cause the routing layer to deviate from the algorithms of LEACH defined in [19], the routing layer is designed in order to achieve maximum fidelity to true life applications, without compromising the results' relevancy to other implementations which may adhere more strictly to the methods outlined in [19].

The first deviation is concerned with LEACH's outlined method of total topological dissolution and reformation at the end of each round. An increasing number of applications have abandoned this approach as they have required the formation of dedicated clusters. This is since, in multi-functional WSNs, such as the one developed for [20], certain tasks are deemed more important than others. In situations such as these, networks are designed with permanent partitioning of nodes into clusters. This is to avoid having nodes assigned to critical tasks depleting their energy by routing messages on behalf of other nodes that were assigned to lesser tasks.

Another deviation is to do with the desirability of multi-hopping features in WSNs. The protocol in [19] defines a method by which nodes communicate with a CH, and then the CH with a sink node, and vice versa. However, in order to allow WSNs to monitor phenomena over very large distances multi-hopping is required. The environmental monitoring deployments of PODs [21], SensorScope [22], and Great Duck Island [23], has found this to be the case, where their large scale deployments were only possible due to the implementation of robust multi-hop networks. Mirroring this requirement, the implemented LEACH exhibits multi-hopping behaviour, while maintaining a hierarchical topology, through the use of nested clusters. To explain, observing Fig.2, node 1, resting in the top layer, is the CH of nodes 2-4, while node 4, acts as the CH of nodes 5-7, making nodes 4-7 a second order, nested cluster.

A third alteration to the LEACH protocols is to do with the CH self-selection process. Since distributing energy level information throughout the network for accurate CH self-selection is a very expensive procedure, synonymous with broadcasting, an alternate approach is suggested. By including energy level information in the packet header, the CH of each cluster is aware of the energy levels of its cluster's members. The CH may therefore accurately carry out the CH selection process by inspecting the headers of its members' data packets, thereby incurring significant savings to the energy consumption and channel occupancy rates of the network.

Finally, the last difference between the implemented and the original protocols is to do with the market-available transceivers. Cheaper transceivers, such as the Nordic Semiconductor's nRF24L01+, do not always have the option of providing accurate RSSI data, operating in a true-mesh design, having an unlimited number of nodes per cluster, or offering varying transmission power levels [24]. The nRF24L01+, for example, suffers from all of these limitations, but, as a trade off, the node sells at one of the lowest prices possible. When combined with the findings of [22], which has shown that the drop in accuracy due to the use of low cost nodes may be counter-acted through the use of many of these nodes, implementations such as [25] have favoured the use of transceivers such as the nRF24L01+ due to their attractive low power and low cost properties.

The routing layer designed therefore, attempts to adapt to all of these limitations, without compromising the applicability of the attained test results. This is done by using a low cost, low power, and highly constrained node, as will be discussed in Section II-C, as well as nested clusters, three packet types, and two operational procedures [20].

The three packet classes used are the 'Data', 'Network Topology', and 'CH Assignment' packet types. The data packet may contain any form of communicable information. The network topology packet is a topological map of all the active nodes within the network. The CH assignment packet is an instructional packet, only transmitted by CHs whenever they have selected an appropriate replacement for themselves in order to inform the selected node that it ought to replace the CH.

All three packets utilize the same header composed of a preamble, link and network layer source and destination addresses, a packet ID, message ID, payload length, acknowledgement (ACK) setting, energy level indicator, packet type identifier, packet fragmentation, and CRC fields.

It is important to note at this point, that in concurrence with the findings of [18], the energy level field only contains a representation of a node's energy level. This representation is in the form of a counter that tallies the number of packets transmitted and relayed by the node in question. Given that the cost of a single packet transmission may be fixed, such as is the case with less versatile transceivers, a packet counter is deemed to be a more useful representation of energy depletion than a constant polling of the battery supply's voltage level.

The operational procedures undertaken by this protocol may be summarized as follows:

1- Once a non-sink node is activated, it periodically transmits data packets to the sink node. The node utilizes CSMA/CA and random back off periods in favour of the hardware and processing needs of more strict time scheduling methods. If the non-sink node is a direct child of the sink node, then it may communicate directly with the sink node. Otherwise, the message must travel through the node's CH. The sink node, having received the message, activates the source's position on its map, before generating a new network topology packet which it then transmits it to every other node in the network. Any node receiving this packet replaces its older map with the newly received version.

2- The CH selection process is initiated after a certain number of packets have been transmitted, representing the expenditure of a certain level of energy. Once initiated, the CH first confirms that it has living children by reviewing its topological map and records of recently relayed messages. Once confirmed, the CH reviews the average energy of its children. If the average is higher than the CH's energy level, then the CH selects the child with the highest energy for the position of CH. Otherwise, if the CH has no children, or the CH's energy is higher than the average, then the CH remains a CH until end of the next round. If a child is selected, then the CH transmits to it a CH assignment message until the message is acknowledged, or a certain number of retries have been met. Once one of these requirements are met, the CH relegates itself to the selected node's position and transmits a message to the new CH. If that message is unacknowledged, then the CH selection process has failed, and the node reverts and repeats the process until it succeeds.

B. *Proposed Fault Tolerance Algorithms*

Due to the limitations of the ways in which the literature has proposed that CH backups, heartbeat messages and self-monitoring be used, a novel approach to adopting the same basic concepts of keep alives and CH backups is introduced. This new method uses three functions to achieve a self-healing system at the lowest possible overhead cost.

The first of these functions uses data packets as keep alives instead of dedicated heartbeat messages. This is since every member of a cluster transmits periodic data packets which must pass through their respective CH. Likewise, these data packets are acknowledged by the relaying CH. Each of these two behaviour are therefore congruent with the concepts of keep alives. Hence, CHs may use data packets as indicators as to which children are alive. This would assist CHs during the CH selection process, in that the CH can now avoid making selections based on old information; as, without this precaution in place, the CH could select and assign the role of CH to a child that had already failed or moved to a different network position. Likewise, by using acknowledgement packets as heartbeat messages as well, unacknowledged packets may therefore be used by child nodes as an indication that their CH has failed. This allows every child to act as a potential CH backup. This is possible once of the CH's children is assigned a number unique to each of their respective network positions. If any of the children then meets a consecutive number of message delivery failures equal to their assigned number, then that child initiates its CH failure functions. A unique number is used to avoid having several nodes initiating their functions at the same time.

The CH failure functions dictate that the node transmits a series of packets to the CH in close succession to each other in order to confirm that the CH has genuinely failed. If these test messages are also unacknowledged by the CH, the node selects one of its children to replace its network position before it rises up to take on the role of the failed CH.

Even though each CH backup is assigned a different number to avoid having several backups initiating CH failure functions at the same time, the possibility still exists. That is, a temporary communication blackout could cause several backups to falsely believe that the CH has failed. This would mean that, eventually, the cluster would in fact have several nodes operating as its CH. A second function to detect and mitigate the existence of duplicate nodes within any cluster is therefore required. This function operates in both a centralized and distributed manner in that the detection of duplicate nodes is the responsibility of the sink node while the resolution of the problem is the task of one of the culprit nodes.

If a sink node receives a message from network position 'A', physical ID '1', and then shortly after, receives a second message from network position 'A', but physical ID '2', the sink node assumes that the network position 'A' is occupied by 2 nodes. The sink node therefore transmits a message to the older source informing it that it is a duplicate node. The node, now aware that it is a duplicate, searches through its topological map for a free position amongst its children and descendants. Once found, the node transmits a test message to that position to confirm that it is in fact free. Once confirmed, the node relegates itself to that position, resolving the problem at hand.

It is important to note that the sink node requires that the older source relegates itself, and not the newer one. This is as this method does not differentiate between the existence of duplicate nodes, and the case where a legitimate handover of a network position through CH selection and assignment may have occurred. If the latter was the case and the sink node targeted newer sources, then the cluster would be left without a CH.

The third and final function attempts to optimize the CH failure mitigation process by better organizing the CH's back-ups. Since each CH backup position requires a different number of consecutive failures to initiate their CH failure functions, the one requiring the least number may be considered the highest backup position. The process may therefore be made more resilient by ensuring that the highest backup position is always occupied by the child with the highest amount of energy remaining. To do so, the CH periodically reviews its topological map and its children's energy levels to ensure that that is the case. However, if the position is found to be unoccupied, then the child with the highest energy is instructed to move to that position. On the other hand, if the position is occupied, but not by the child with the highest energy, then the two nodes are instructed to switch positions.

C. Hardware and Test-Bed Design

The physical nodes designed are developed within the same low power and low cost goals and constraints as the overall network. Each node is made up of an nRF24L01+ transceiver, an ATMega328-PU microcontroller, a 10K pull-up resistor, two 22pF capacitors and a 16MHz crystal for clocking. This design costs 10 USD and, when operated at 3.3V, utilizes 7.5-8.0 mAh, achieving the aims of developing for a low power, low cost WSN. A total of 49 of these nodes were loaded on five breadboards with two further breadboards used as power rails to make up the entire test bed shown in Fig. 1 [20].

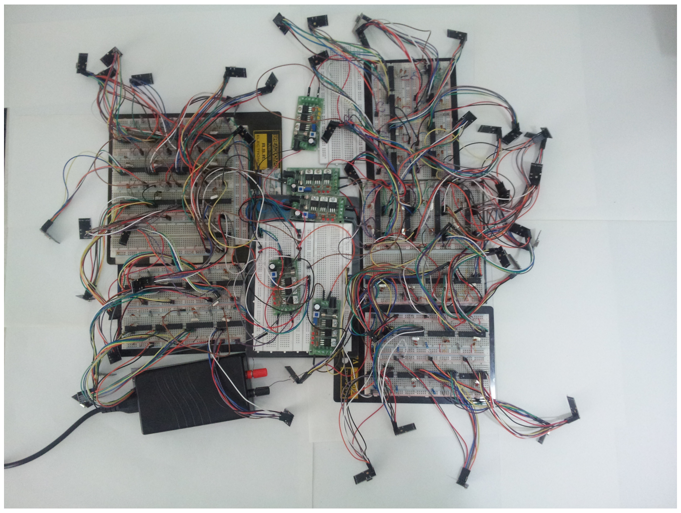


Fig. 1: The Hardware Testbed [20]

Each node is instructed to halt once a certain number of packets have been transmitted. At that point, the node is expected to record all of its results on its internal EEPROM and awaits their manual collection.

The results recorded includes the total number of successful and failed data and overhead packets transmitted, all the network positions occupied by the node, the reason it halted, and the amount of energy the node had when it halted.

In addition to the end-of-sim data, an in system programmer (ISP) is always connected to the sink node to provide real time coverage of all network happenings. An ISP may also be connected to any other node in the network in order to observe the network's operations from its point of view. The data obtained through this kind of monitoring includes the time that each packet is received, acknowledged, and transmitted by the monitored node, as well as the packet's type, physical ID and network ID source and destination addresses, and the results of any initiated CH selection, assignment, or fault tolerance function.

In order to observe the performance of the proposed algorithms under duress, several faulty nodes are programmed for testing purposes. As described in Section I, a node suffering from intermittent drops in its link quality may still be selected as a CH, and may still have nodes join its cluster. In order to model this behaviour, several nodes have been programmed to transmit and receive, but to refuse to relay messages on behalf of other nodes. This way, the node may still participate in CH selection processes, and act on CH assignment messages, while maintaining the behaviour expected of a CH that has a low quality channel connecting it to its children.

III. System Evaluation and Results

A. Test Behaviour and Testing Parameters

A total of four tests were performed in order to explore the effectiveness of the developed fault tolerance algorithms. As required by LEACH, the first two tests used data packets to

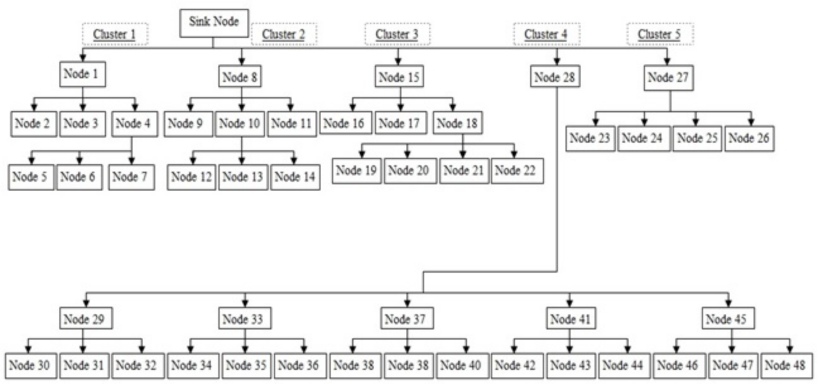


Fig. 2: The network topology of tests 1-4 [20].

communicate each node's energy level information to all other nodes in the network. The last two tests, however, examine the approach proposed in Section II-A, whereby energy level information is included in packet headers.

The first test observed the effect of having only the CH failure detection and mitigation techniques. The second test added the duplicate node detection and mitigation functions, and one faulty node. The third test repeated test two using the modified packet header and two faulty nodes. The fourth test added the CH backup optimization functions to test three.

All four tests utilized the network topology shown in Fig. 2, which was designed as such to observe the effect of having different node distributions [20]. Cluster 1 had a significant amount of energy associated with a lower CH backup position than cluster 2. Cluster 3 had an increased number of nodes at the third layer. Cluster 4 had a much larger size than all of the other clusters and, finally, cluster 5, was composed of only 2 layers of nodes.

Each of the four tests were run a total of five times, with each node being given 2000 packet tokens per iteration. The CH selection and assignment functions were initiated with every 50th packet transmission. The inter-packet transmission time for every node was set to (2*N) seconds, where N was the total number of nodes in the network.

The above testing parameters allowed for the collection of over 480 hours of simulated runtime. The large amount of data collected caused us to represent this data in the summarized forms of Tables I-IV and Fig 3. Table I distributes the nodes of each test into success rate intervals based on their data packet delivery success rates. Table II attributes to each failure type observed the percentage of the total number of failure messages for which it was responsible in each respective test. Table III gives an overview of the times at which significant network failures occurred, and Table IV shows the average overhead required to operate the implementations of each of tests 3 and 4. Lastly, Fig. 3 shows the rate of energy token depletion for each node and for each cluster, for both of tests 3 and 4.

B. Test Results

Test 1 observed the effect of adding only the CH failure detection and mitigation functions to the routing layer of Section II-A. The results show only 40.7% of nodes achieving over 90% success rates, while the bulk of nodes ranged between 50-80% success rates. This was due to the false detection of CH failures allowing for the appearance of duplicate nodes. After the first case of duplicate nodes appeared, any message passing through their network position was replicated before exiting from both nodes causing an unnecessary strain on the network's energy resources. Also, as the nodes would acknowledge messages destined for their network position at the same time, the ACK packets would interfere with each other such that the message source would believe that the message delivery failed. This would cause the source to retransmit and burden the network's already limited resources even further. Also, as the duplicate nodes would consistently interfere with each other's ACK messages, the children would then falsely believe that their CH has failed, causing them to rise up to the role of CH, meaning that the issue of duplicate nodes is in fact self-perpetuating in nature.

The kind of behaviour described above directly caused over 50% of the recorded packet failures, as well as significantly impacting the cluster's lifetimes. Clusters found to suffer from larger numbers of duplicate CHs, such as clusters 3 and 5, had longer lifetimes than the other clusters. This was as these cluster had fewer children and so spent less energy relaying messages in comparison to other clusters. Instead, the bulk of the packet tokens were used up at the much slower pace of generating and transmitting new packets. Also, the increased number of replicated messages exiting these clusters caused the destination clusters to have much shorter lifetimes as they would carry most of the burden of having to route these messages.

Another factor impacting node lifetimes was LEACH's requirement that energy level information be transmitted across the network for accurate CH selection and assignment. This behaviour was found to cause a very early first death in the network. This was because, given the inter-packet transmission time, and the long list of nodes that had to be informed, the dissemination of energy level information was a very slow process. Effectively, nodes assigned to CH positions would rarely receive this information quickly enough to avoid dying due to the intense energy needs of it being a CH. This elimination of nodes through over-exhaustion was found to continue until the number of nodes within each cluster was reasonable enough to allow for the distribution of energy level information at a speed ample enough to permit each cluster to run effective CH selection and assignment processes. In addition to causing a very early first death, this also allowed cluster 4 to have a significantly larger lifetime than all other clusters. This is as, by the time that the network had shed off many nodes to operate efficiently, cluster 4 would still have the most number of nodes remaining, and hence, the largest pool of energy left to spend.

Starting with the problem of duplicate nodes, test 2 attempted to limit their considerable impact by introducing the duplicate nodes detection and mitigation functions to the network's routing layer. The test was performed while having a single faulty node occupying the position of node 4.

Test 2 observed a reduction in the overall success rates of the network. Around 11.0% of nodes had less than 50% success rates. This was in equal parts due to the duplicate mitigation functions, the faulty transceiver, and the overhead used to operate the routing layer. To explain, the faulty transceiver, being the only node that can transmit and receive, but not relay messages, would always be the node with the highest energy. This meant that the CH selection process would always assign it the role of CH. This would cause all of its children to detect message delivery failures causing them to believe that their CH has failed. Consequently, a node would rise up to replace a CH which has not failed, and two nodes would be occupying the same position. The sink node would then have to inform the faulty node that it should relegate itself, causing the faulty node to initiate its duplicate node mitigation functions. This process repeats continuously meaning that there a persistently large amount of overhead was constantly travelling through the network. This, in addition to the amount of overhead required to circulate energy level information, and confirm that CHs have failed or that network positions were free, meant that the majority of failures experienced in the network were mainly due to this overhead, causing 72.16% of packet failures, and 59.6% of nodes to lie within the 50% to 80% success rate interval.

Another cause of failure observed during these tests is associated with nodes operating within the fourth layer of a cluster. This type of failure was responsible for 10.2% of the failed messages. Although nodes were never instructed or programmed to operate within the fourth layer of any cluster, occasionally, the duplicate mitigation function would fail to find an empty position within the top three layers for the node to relegate itself to. The node would therefore relegate itself to the fourth layer of the cluster. As expected from the findings of [21] and [23], operating at this distance from the sink node incurred an increased level of message delivery failures.

In terms of failure times, the slow spread of energy level information caused an early first death for the same reasons as test 1. However, in this case, the efficient resolution of duplicate nodes by the included duplicate mitigation functions allowed test 2 to have much more equal cluster failure times.

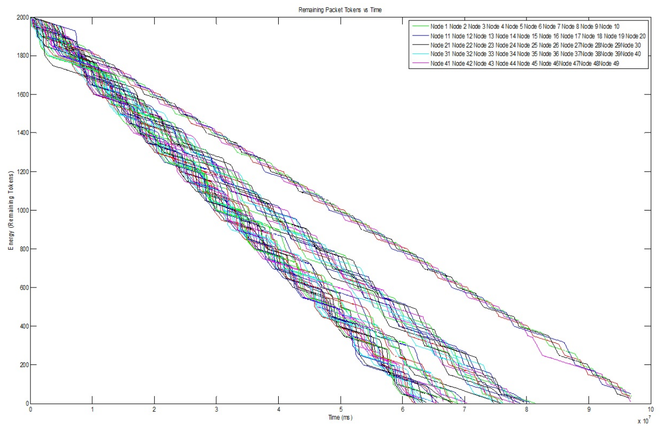
In order to reduce the overhead and quicken the propagation of energy level information throughout the network, test 3 had energy level information included in packet headers. This significantly reduced the overhead required to operate the network. This meant that even in the presence of two faulty nodes, started at positions 4 and 37, 95.4% of nodes achieved a success rate of over 90%. This method also meant that each cluster was able to distribute the energy consumption rates of all its nodes in a more balanced way, as shown in Fig. 3(a) and Fig. 3(c), as the required information was now more readily available. This allowed the network to have improved values for the recorded lifetimes all around, with the most notable difference being that the first node failure occurred at almost 7 times the value attained in test 2.

The increase in the CH selection and assignment's effectiveness also meant that faulty nodes were quickly isolated away from any critical roles in the network. Although they were still selected as CHs, the increase in the network's efficiency meant that these nodes would spend very little time as CHs before receiving duplicate mitigation packets, quickly causing them to be relegated to the lowest layers of the network. This meant that their effects on other nodes were effectively curtailed, as exemplified by the low percentage of nodes achieving less than 90% success rates.

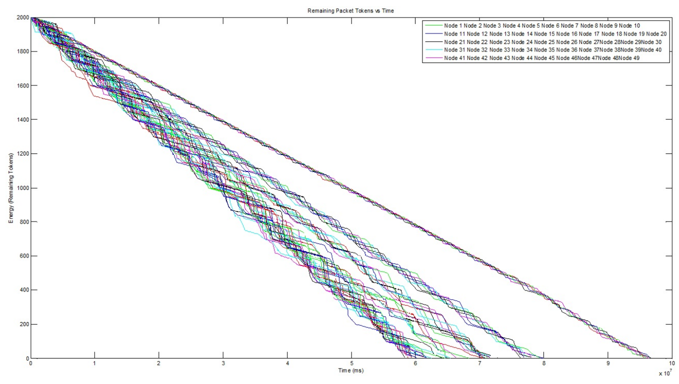
Exploring the failure times further, clusters 1, 2 and 3 had fairly similar lifetimes. Clusters 1 and 3, having the bulk of their backup energy at the lowest position, achieved lower lifetimes than cluster 2. Cluster 3, on the other hand, outlived cluster 1 due to the energy added to it by including a single extra node. This however is contrasted by the lifetime of cluster 4, which had the shortest lifetime of all. This was because, by having the largest number of nodes in a properly operating network, cluster 4 was burdened with having to generate and route the largest amount of messages out of all the clusters. Cluster 5, operating at only 2 layers, spent the least amount of energy on relaying, and therefore expended its tokens at the much slower pace of transmitting original packets, causing it to outlive all of the other clusters.

In an effort to further improve the success rates achieved by test 3, test 4 repeated the iterations of test 3 while including the CH backup optimization functions. Although expected to extend the network's lifetime and improve data success rates, what was in fact observed was an increase in the overhead used, which was found to be several times the amount required by the implementation of test 8. The network was found to be predominantly concerned with maintaining a specific structure rather than achieving high data delivery rates. This had a negative impact on the network's success rates without any discernible gain in terms of the network's lifetime.

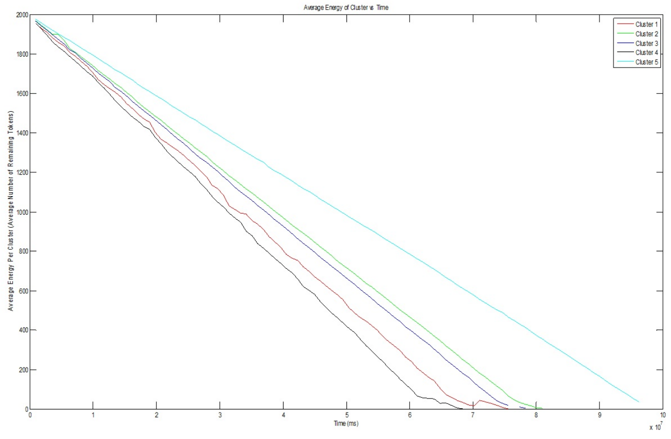
To elaborate, the decrease in the success rates was primarily



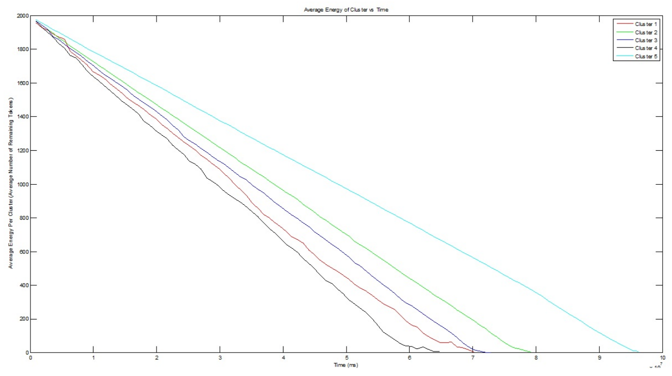
(a) Test 3



(b) Test 4



(c) Test 3



(d) Test 4

Fig. 3: The rate of depletion of the energy tokens of the nodes and clusters of tests 3 and 4.

TABLE I: Distribution of Nodes into Success Rate Intervals

	Percentage of Nodes Per Success Rate Interval (%)			
Test #	>90%	80-90%	50-80%	<50%
1	40.7	31.4	25.7	2.1
2	6.5	22.9	59.6	11.0
3	95.4	2.9	1.7	0
4	79.2	17.0	3.8	0

TABLE II: Percentages of Failed Messages vs. Failure Cause

Failure Type	**Test 1**	**Test 2**	**Test 3**	**Test 4**
Scheduling	9.82%	0.25%	31.03%	15.55%
Overhead	37.63%	72.16%	15.75%	33.81%
Transceiver	0%	17.40%	53.22%	50.65%
Layer 4	0%	10.19%	0%	0%
Duplicates	52.55%	0%	0%	0%

due to two specific reasons. First, the elevated levels of overhead packets traversing the network made it more difficult to acquire channel time, and increased the chances of mid-

TABLE III: Average Failure Times (Hours:Minutes:Seconds)

Failure Type	**Test 1**	**Test 2**	**Test 3**	**Test 4**
First Failure	00:44:22	01:36:36	10:56:56	12:04:13
Cluster 1	15:24:21	17:27:06	20:34:20	20:21:55
Cluster 2	12:21:17	15:29:56	22:26:45	22:04:45
Cluster 3	26:21:26	18:07:27	21:55:19	20:38:45
Cluster 4	35:53:55	18:08:15	19:32:21	19:00:22
Cluster 5	25:53:39	14:58:39	27:04:04	26:09:21
Failure of Last Living Node	36:15:17	21:00:38	27:04:04	26:09:21

TABLE IV: Average Overhead (% of Total Number of Packets Transmitted by a Cluster)

Cluster	**Test 3**	**Test 4**
Cluster 1	8.4%	33.8%
Cluster 2	5%	11.2%
Cluster 3	5.8%	14.8%
Cluster 4	6.6%	23.6%
Cluster 5	4.8%	11.8%
Total	6%	19.6%

air collisions. Second, the selection of the faulty nodes in clusters 1 and 4 as the primary backup, as they constantly had the highest energy levels available, once again amplified their effects on the network's success rates.

It is important to note that even with these two inhibiting factors, Fig. 3(b) and Fig. 3(d) show that the implemented protocol is able to withstand their effect, as exemplified by the stable and distributed rates of energy token depletion across each respective cluster.

Other than the aforementioned observations, all other network performance values observed, such as percentages of failed messages due to scheduling and the faulty nodes, and network failure times, were very similar to those of test 3 as no other model principles were modified in test 4 to impact these variables.

IV. Conclusion

A mixed method approach at achieving fault tolerance, detection and mitigation is proposed for a highly constrained implementation of LEACH. For a minimal amount of overhead, the described algorithms allow for high data delivery success rates while addressing several flaws in the original designs of LEACH, as well as those found in the fault tolerance methods advised for LEACH by recent literature.

The approach is tested using a hardware basin atop which a multi-hopping network of permanent clusters is built in order to test the proposed methods in a manner relevant to the needs of recent applications. The development and subsequent use of low cost, low power nodes of low capabilities attest to the methods' ability to guarantee the efficient dissemination of energy level information, the isolation of non-performing nodes to non-critical roles, and the gradual depletion of node's energy stores, all within the constraints of bare minimum designs.

The study also verifies that multi-hopping should be limited to 3 hops from the sink node at most. This is as the deployment of any more than 3 layers of nested nodes causes a severe increase in the number of messages which failed to reach their destination, as well as a significant decrease in cluster lifetimes.

Lastly, the research proves that the semi-centralized approach suggested for the optimization of CH backups' distribution is too costly of an approach for the given deployment, without any apparent added benefit.

References

[1] S. Tyagi, N. Kumar, "A systematic review on clustering and routing techniques based upon LEACH protocol for wireless sensor networks", J. of Network and Computer Appliances, vol. 36, pp. 623-645, Dec 2012.

[2] L. de Souza, H. Vogt, M. Beigl, "A Survey on Fault Tolerance in Wireless Sensor Networks, SAP Research, Karlsruhe University, Germany, 2007.

[3] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin, "Sympathy for the Sensor Network Debugger", Proc. of the 3rd International Conference on Embedded Networked Sensor Systems, 2005.

[4] Y. J. Zhao, R. Govindan, and D. Estrin, "Residual Energy Scan for Monitoring Sensor Networks, IEEE Wireless Communications and Networking Conference, vol . 1, pp. 356-362, 2002.

[5] S. Dawson-Haggerty, A. Krioukov, J. Taneja, S. Karandikar, G. Fierro, N. Kitaev, and D. Culler, "BOSS: Building Operating System Services", Proc. of the 10th USENIX Symposium on Networked Systems Design and Implementation, pp. 443-457, 2013.

[6] M. Turon, "MOTE-VIEW: a sensor network monitoring and management tool", Proc. of the 2nd IEEE Workshop on Embedded Networked Sensors, pp. 11-17, 2005.

[7] M. M. Alam, M. Mamun-Or-Rashid, and C. S. Hong, "WSNMP: A Network Management Protocol for Wireless Sensor Networks", Proc. of the 10th International Conference on Advanced Communication Technology, vol. 1, pp. 742-747, Feb. 2008.

[8] R.V. Kshirsagar and B. Jirapure, "A Survey on Fault Detection and Fault Tolerance in Wireless Sensor Networks, Proc. of the 1st International Conference on Benchmarks in Engineering Science and Technology, pp. 6-9, Oct. 2012.

[9] N. Ramanathan and M. Yarvis, "A Stream-oriented Power Management Protocol for Low Duty Cycle Sensor Network Applications", in Proc. of the 2nd IEEE Workshop on Embedded Network Sensors, May 2005.

[10] J. Zhang, E.C. Kulasekere, K. Premaratne, and P.H. Bauer, "Resource Management of Task Oriented Distributed Sensor Networks", in Proc. of the IEEE International Symposium on Circuits and Systems, vol. 2, pp. 513-516, 2001.

[11] A. Boulis and M.B. Srivastava, "Node-level Energy management for Sensor Networks in the Presence of Multiple Applications", in Proc. of the 1st IEEE International Conference on Pervasive Computing and Communications, pp.41-49, Mar. 2003.

[12] M. Perillo and W.B. Heinzelman, "Providing Application QoS through Intelligent Sensor Management", in Proc. of the 1st IEEE International Workshop on Sensor Network Protocols and Applications, 2003.

[13] A. Akbari, N. Beikmahdavi, A. Khosrozadeh, O. Panah, M. Yadollahi, and S.V. Jalali, "A Survey Cluster-Based and Cellular Approach to Fault-Detection and Recovery in Wireless Sensor Networks", World Applied Sciences Journal, vol. 8, No.1, pp. 76-85, 2010.

[14] A. Akbari, N. Beikmahdavi, and M. Mohammadi, "A New Algorithm Fault Management by Clustered in Wireless Sensor Network, World Applied Sciences Journal, vol. 12, No. 10, 2011.

[15] A.S. Mohammed, and M. N. Shanmukhaswamy. "New Algorithm for Optimized Cluster Heads with Failure Detection and Failure Recovery to Extend Coverage of Wireless Sensor Network, International Journal of Scientific and Research Publications, vol. 2, No. 11, Nov. 2012.

[16] G.V. Selvi, and R. Manoharan, "Cluster Based Fault Identification And Detection Algorithm For WSN A Survey. International Journal of Computer Trends and Technology, vol. 4, No. 10, 2013.

[17] M.H. Yin, and Z. Win, "Fault Management Using Cluster-Based Protocol in Wireless Sensor Networks. International Journal of Future Computer and Communication, vol. 3, No.1, Feb. 2014.

[18] J. Zhang, S. Ci, H. Sharif, and M. Alahmad, "Modeling Discharge Behaviour of Multicell Battery", IEEE Transactions on Energy Conversion, vol. 25, No.4, pp. 1133-1141, Dec. 2010.

[19] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks, Proceedings of the Hawai'i International Conference on System Sciences, Maui-Hawai'i, Jan. 2000.

[20] A. Ismail, and K. Seddik, "The Design and Implementation of a Constrained WSN for Permaculture Farming in Egypt", To be Published in the Proceedings of 19th IEEE International Conference on Emerging Technologies and Factory Automation, Sept. 2014.

[21] E. Biagioni, "Pods: Issues in the Design of Practical Ad-Hoc Wireless Sensor Networks", SSGRR Summer 2003 Conference, 2003.

[22] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "SensorScope: Application-Specific Sensor Network for Environmental Monitoring", ACM Transactions on Sensor Networks, Vol. 6, No. 2, Article 17, pp. 1-32, Feb. 2010.

[23] J. Polastre, R. Szewcyk, A. Mainwaring, D. Culler, and J. Anderson. Analysis of wireless sensor networks for habitat monitoring. In Raghavendra, Sivalingam, and Znati, editors, Wireless Sensor Networks, pages 399423. Kluwer Academic Pub, 2004.

[24] Nordic Semiconductor. "nRF24L01 Single Chip 2.4GHz Transceiver Product Specification", July 2007.

[25] L. Urdiain, C. Romero, J. Doggen, T. Dams, and P. Van Houtven, "Wireless Sensor Network Protocol for Smart Parking Application: Experimental Study on the Arduino Platform, Proceedings of the 2nd International Conference on Ambient Computing, Applications, Services and Technologies, Sept. 2012.