Fault-Tolerant Clustering of Wireless Sensor Networks

Gaurav Gupta
Dept. of Computer Science and Elec. Eng.
University of Maryland Baltimore County
Baltimore, MD 21250
gagupta1@cs.umbc.edu

Mohamed Younis
Dept. of Computer Science and Elec. Eng.
University of Maryland Baltimore County
Baltimore, MD 21250
younis@cs.umbc.edu

Abstract-- *During the past few years distributed wireless sensor networks have been the focus of considerable research for both military and civil applications. Sensors are generally constrained in on-board energy supply therefore efficient management of the network is crucial to extend the life of the system. Sensors' energy cannot support long haul communication to reach a remote command site, thus they require multi-tier architecture to forward data. An efficient way to enhance the lifetime of the system is to partition the network into distinct clusters with a high-energy node called gateway as cluster-head. Failures are inevitable in sensor networks due to the inhospitable environment and unattended deployment. However, failures in higher level of hierarchy e.g. cluster-head cause more damage to the system because they also limit accessibility to the nodes that are under their supervision. In this paper we propose an efficient mechanism to recover sensors from a failed cluster. Our approach avoids a full-scale re-clustering and does not require deployment of redundant gateways.*

Keywords: Network clustering, Fault-tolerance, Energy-Aware Communication, Sensor networks.

1. Introduction

Recent advancements in integrated circuits have fostered the emergence of a new generation of tiny, inexpensive low-power sensors. Due to their economic and computational feasibility, a network of hundreds and thousands of sensors has the potential for numerous applications in both military and civil applications such as combat field surveillance, security and disaster management. These sensing devices are capable to monitor a wide variety of ambient conditions such as: temperature, pressure, motion etc. The sheer number of these devices and their ad-hoc deployment in the area of interest brings numerous challenges in networking and management of these systems. Sensors are typically disposable and expected to last until their energy drains. Therefore, energy is a very scarce resource for such sensor systems and has to be managed wisely in order to extend the life of the sensors for the duration of a particular mission.

Typically sensor networks follow the model of a command node or base station, where sensors relay streams of data to a command node either periodically or based on events. The command node is located faraway from the area where the sensors are usually deployed. In order to conserve energy consumed in communication with the command node various multi-hop and energy aware routing techniques have been suggested in the literature [5][6]. These techniques have overhead due to route discovery and to find optimum hops to communicate with the command node. In addition, there will be extra burden on the nodes, which are located around the command node, as most of the traffic will be routed through them.

To avoid these overheads and unbalanced consumption of energy some high-energy nodes called *"Gateways"* are deployed in the network. These gateways, group sensors to form distinct clusters in the system, manage the network in the cluster, perform data fusion to correlate sensor reports and organize sensors by activating a subset relevant to required missions or tasks as shown in Fig 1. Clusters are formed based on the load on the gateways and the communication distance between sensors and the gateways

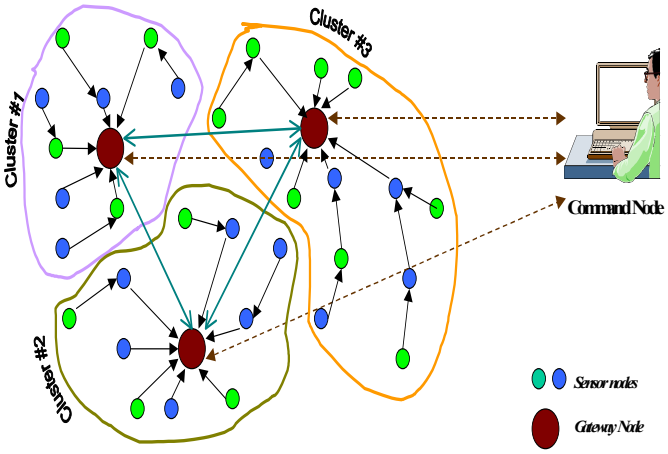

Fig. 1: Multi-gateway clustered sensor network

[8]. Each sensor belongs to only one cluster and communicates with the command node only through the gateway of the cluster.

In sensor networks the effectiveness of data fusion depends not only on the sensed data but also on the coverage of sensors. In some mission critical applications such as disaster management it is essential to ensure good coverage to increase the potential of rescuing survivals and ensure the safety of the rescue crew. Therefore, dependability of the system becomes another very important factor for the efficient operation of the system. Sensors are susceptible to

device failures due to limited battery power but will also be inactive if the gateway in their cluster suffers from some faults. Reconfiguration of the system can be used to recover the sensors in a faulty cluster through re-clustering. Re-clustering the system complicates the network setup and bootstrapping. Gateways have to stop data processing and communication in order to perform clustering. New communication schedules have to be set and transmitted to the sensors. Moreover, frequent faults will result in frequent re-clustering wasting precious energy and time. Redundant gateways can also be deployed in the system to replace the faulty gateways. However, pre-deployment of redundant gateways makes them unutilized resource while replacement of faulty gateway can be impractical and slow.

In this paper we investigate the dependability of sensor networks in the presence of faults in the gateways. We propose a run-time recovery mechanism based on consensus of healthy gateways to detect and handle faults in one faulty gateway. A two-phased detection and recovery mechanism is proposed to limit the performance impacts caused by a gateway failure. We use a simulation-based fault injection method, which assumes that errors occur according to a predetermined distribution. The sensors assigned to the faulty gateway are reorganized on the fly without bringing the system to a complete shutdown. The recovery information is created during clustering which facilitates the recovery process. Various communication fault scenarios are considered and handled during recovery. Our approach provided considerable improvement in the stability of the system and reduces the overhead of re-clustering and system reconfigurations.

In the next two sections we define the architectural model of sensor networks and summarize the related work. Section 4 describes the fault-detection and recovery approach. Description of the simulation environment and validation of the experiments can be found in section 5. Finally section 6 concludes the paper and discusses our future research plan.

2. System Model

The system architecture for the sensor network is shown in Fig 1. There are only two kinds of nodes in the system; sensors and less-energy-constrained gateway nodes. The sensors and gateways are assumed to be of the same kind and have same properties respectively. All communication is over wireless links. A wireless link is established between two nodes only if they are in range of each other. Gateways are capable of long-haul communication compared to the sensors and are in direct communication range with the command node. Communication between nodes is over a single shared channel. Current implementation supports TDMA [7] protocol to provide MAC layer communication.

In this paper we assume that the sensor and gateway nodes are stationary. In the future we plan to incorporate mobile gateways in the system. During the bootstrapping process, all the sensors and gateways are assigned unique IDs, initial energy and TDMA schedule. All nodes are assumed to be aware of their position through some GPS system. While the GPS consumes significant energy, it has to be turned on for a very short duration during bootstrapping. Sensors inform the gateways about their location during the clustering process. It is worth noting that most of these capabilities are available on some of the advanced sensors, e.g. the Acoustic Ballistic Module from SenTech Inc. [2].

Initially all gateways are assumed to be in communication range with one another. Gateways form their own subnet to exchange status information about the clusters and to reach a consensus during recovery. The schedule of first inter-gateway communication is known to all the gateways during bootstrapping. No communication between the gateways and sensors is scheduled during inter-gateway communication.

2.1 Fault Model

A system failure occurs when the delivered service deviates from the specified service [17]. Hardware and software faults affect the system state and the operational behavior, such as memory or register content, program control flow, and communication links etc. We assume a fail silent model where any erroneous behavior does not affect the healthy components. We assume that the communicated data is error free and semantic-related generic faults in the software are detected and removed by application-specific checks.

Communication faults can be caused due to hardware failure or energy depletion. Communication can be disrupted due to environmental conditions like wind or rain. Hardware faults can also disrupt radio communication, ending all the communication to and from the gateway. A fault in transmitter can prevent the gateway to transmit tasks to the sensors as well as relay the data to the command node. Data send by the sensors will be lost if receiver of a gateway fails. We call all such failures as complete gateway failures because the gateway can no longer serve as a liaison between the sensors and the command node. Another kind of failures is caused due to faults in range of gateway. Faults in range of the device can affect its coverage. A gateway can experience communication link failure between the sensors in its cluster or with other gateways. A communication link failure with the sensors requires the sensors to be allocated to other gateways within communicate range. Faults in inter-gateway communication are handled through forwarding approach explained later.

Based on the temporal behavior of a fault it can be considered as permanent, intermittent or transient. In our fault model we consider only permanent faults. A permanent fault once activated remains effective until it is detected and handled. We also assume that the system is not liable to Byzantine-type faults [20].

3. Related Work

Our work is motivated by a various research projects in sensor network domain. Researchers are exploring both

hardware and software aspect of sensor networks. Projects like Smartdust [9], WINS [10], PicoRadio [11] have given a new dimension to the size and capabilities of sensors. Since sensors are typically battery-operated with limited energy supply, many research groups have focused on issues like energy aware routing [5], sensor coordination [6], and energy saving through activation of a limited subset of nodes [4][12].

Many clustering approaches have been proposed for efficient selection of a cluster-head such as randomized [13] lowest cluster-ID [15], or highest degree of connectivity [16, 17]. However, if load is not balanced among the cluster it can lead to increased latency in communication, inadequate tracking of targets or events and finally results in failure of the gateway. In our previous work a multi-gateway architecture is presented to cluster the network around high-energy gateways while balancing load among the clusters [8].

Moreover, these approaches do not focus on dependability and fault-tolerance in the system. Upon failure of a cluster-head either the role is reassigned to another node requiring re-configuration of the whole system or redundant hardware is used as replacement. Projects like LEACH [13] include redundancy in the system by periodically selecting a cluster-head from the sensors in the network but suffer from overhead of re-clustering. We believe that, significant performance gain can be achieved if efficient recovery is embedded in the system from the beginning. Faults should be detected and handled during the run-time. Analysis and modeling of faults is a well-researched field [20]. In this paper we present a run-time recovery mechanism, which detects faults in gateways and recover sensors from the failed clusters.

4. Fault-Tolerance Mechanism

The main objective of our approach is to perform run-time recovery of the sensors from the clusters in which the gateway has experienced some faults. The mechanism is divided in to two phases; detection and recovery. In order to recover the sensors from the failed cluster it is important to detect whether a fault has occurred in the system. We follow a consensus model of the gateways to agree on a particular fault in the system. A consensus is required to maintain the synchronization in the network with respect to the status and cardinality of a gateway. The cardinality of a gateway is the number of sensors that belong to the cluster of a gateway. In later sections we present scenarios where gateways can have conflicting knowledge about the status of a gateway and explain methods to avoid it. The second phase of fault-tolerance identifies the type of fault and performs recovery of the sensors.

4.1 Detection of gateway failure

Detection is the first phase of fault-tolerance in our system. All the gateways in a sensor network are independent identities. A gateway is responsible only for the sensors in its own cluster. We adopt a method of periodic status updates through inter-gateway communication. Status updates inform all the gateways about the whereabouts of the rest of the clusters in the system.

As mentioned in section 2, we are using TDMA MAC protocol for communication. TDMA schedules for sensors are decided by their respective gateways. Typically, gateways allocate slots for sensors to send data based in available energy, tasks, and priority [7]. Fig 2 shows a simple slot allocation for a gateway. Sensors are informed about the schedule and routing information in a "Route Update" slot. The dark slots represent the route update slots and the white slots are reserved for sensors to send data in that cycle. Along with the sensed data, sensors also provide their energy status to the gateways. A cycle is completed when all the sensors send data and energy status to their respective gateways and wait for the next route update. At the end of every cycle each gateway constructs a "Status" containing information about the sensors in its cluster and the status of the gateways itself.

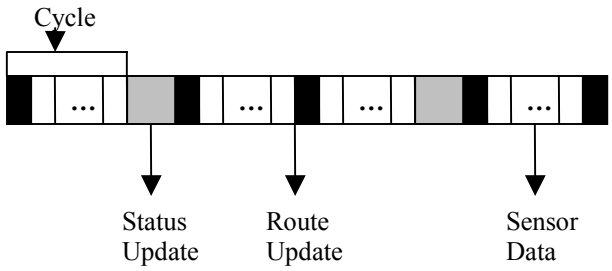

Fig. 2: Slot Allocation in Sensor Networks

Gateway status is exchanged in a "Status Update" slot (shown as grey slots in Fig. 2) whose period depends on the stability of the system. We use a Multiplicative Increase Linear Decrease (MILD) mechanism to schedule the status exchange. In the absence of faults, MILD increases the time period of the exchange by a multiplicative factor while linearly decreases the time period when a fault is detected. By this method we reduce the overhead of status exchange when the system is stable and recovers fast from the faults when the system is fragile. Status messages also act as heartbeat messages from the gateways informing about their presence. At the end of detection phase when a gateway "A" does not receives update from another gateway "B", gateway "B" is considered to be faulty by "A". Since the updates can be missed due to link failures between two nodes, a consensus has to be reached by all gateways before recovery commences. It is important to remember that a gateway should not be considered completely failed until even one of the gateways in the network is able to communicate with it.

In case of link failures multiple hops have to be used to forward updates. Efficient routing can be used to forward these updates but they require maintenance and update of routing tables. For the purpose of this paper we adopt a simple forwarding approach. Each gateway forwards (broadcasts) every "*new*" update it receives to all the gateways in its range. This method will add redundant messages in the network when the network is fault-free but

ensures that every gateway has the same status information of the system. A consensus is reached automatically since all the gateways share the same information. If a gateway has failed none of the other gateways will receive the update and can start the recovery. We describe two scenarios to explain the forwarding approach and introduce an experience-based enhancement to avoid redundant messages in the absence of faults.

Case 1: No Faults, fully connected network

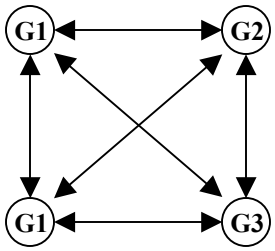

Fig. 3: Fully Connected Gateway Model

Fig 3 shows a fault free fully connected gateway architecture where all gateways (G1-G4) are in direct communication range with one another. During the status update phase all the gateways will broadcast their updates. Since all the gateways are in direct communication range every gateway will receive the status and will conclude that no gateway has completely failed in the system. But, the forwarding algorithm will make the gateways broadcast the redundant status information of other gateways as well.

In order to avoid such message redundancy in the absence of faults in the system we use an *"experience"* based model. Before forwarding updates from other gateways each gateway constructs an experience of the updates received. They first broadcast their experience about the connectivity with other gateways. After receiving the experience from other gateways, an experience table is constructed that shows the connectivity of different nodes in the system. For the case described above the experience table is shown in Table 1. When a gateway receive the experiences like the one shown in table below, it signifies that the network is fully connected and no forwarding of update message is required.

	G1	G2	G3	G4
G1	√	1	1	1
G2	1	√	1	1
G3	1	1	√	1
G4	1	1	1	√

Table 1: Experience Table for Case 1

Where:
√ signifies own update
1 signifies that the update is received
0 signifies that the update is missed

Case 2: Multiple link failure and single complete failure

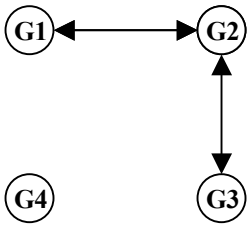

Fig. 4: Multiple Link and Single Complete failure Model

Fig 4 shows the system architecture after link failures between gateway G1 and G3 followed by a complete failure of gateway G4. In the first status update slot no gateway will receive status from G4. Also, G1 and G3 will not receive status from each other and G2 will receive status from both G1 and G3.

	G1	G2	G3	G4
G1	√	1	0	0
G2	1	√	1	0
G3	0	1	√	0
G4	0	0	0	0

Table 2: Experience Table for Case 2

The experience table formed at gateway G2 is shown in Table 2. After analyzing the experience table, gateway G2 realizes that none of the gateways has received status update from gateway G4 and G4 had not send its experience to any of the gateways. This clearly indicates that G4 is not able to transmit any data to other nodes due to transmitter fault. Therefore, G4 is tagged as completely failed and all the sensors in the cluster of G4 have to be recovered.

The zeros in the experience on G1 and G3 indicate the link failure between them. Since a consensus cannot be reached about the complete failure of Gateway G4 unless all the gateways receive the experience, G2 understand that it has to forward the update to G1 and G3. Once the gateways G1 and G3 receive all the update except from G4 they also concur to the complete failure of gateway G4.

4.2 Recovery

Once the gateways reach a consensus about the presence of a fault, the next step is to identify the type of faults and allocate the sensors to new clusters. The status message is parsed to extract the identity of sensors that cannot communicate with the gateway due to range faults in the gateways. When a gateway is identified as completely failed all the sensors in its cluster are recovered.

Clustering is based on the distance between the sensors and gateways. During clustering each gateway creates a range set based on the communication range of the sensors and the gateways. A sensor 'S_j' belongs to range set 'RSet' of gateway 'G_i' if it satisfies the following criteria:

$$S_j \in RSet_{Gi} \Leftrightarrow [(R_{Gi} > d_{Sj->Gi}) \wedge (R_{Sj,max} > d_{Sj-->Gi})]$$

Where, R_{Gi} is the range of gateway G_i, $R_{Sj,max}$ is the maximum range of sensor S_j and $d_{Sj->Gi}$ is the distance between sensor S_j and Gateway G_i. A final set (FSet) is constructed based on the minimum communication cost between sensors and gateways [8]. For the purpose of recovery each gateway constructs another set containing nodes that do not belong to the cluster of the gateway but are included in its RSet. This set is called a Backup set (BSet). Each node only belongs to a single FSet but can be part of many BSets. The definition of BSet is defined as:

$$S_j \in BSet_{Gi} \Leftrightarrow [(S_j \in RSet_{Gi}) \wedge (S_j \notin FSet_{Gi})]$$

When a sensor has to be recovered all the gateways check their own BSets for the sensor. The sensor is recovered if it is present in the BSet of the gateway. If a sensor is present in multiple BSets, it is accommodated by the gateway, which has the minimum communication cost with the sensor other then one failed. Once the sensor is associated with the backup gateway, it is removed from the BSet of the backup gateway as well as the RSet of the faulty gateway.

Due to previous schedule the receivers of the sensor are turned on during the route update slot to receive the new update from the gateway. Therefore, the backup gateway informs the sensor about the new association in the same slot. New TDMA schedules are given to the sensor according to the cardinality of the new gateway and the sensor becomes a part of the backup cluster.

5. Experimental Validation

The effectiveness of our recovery approach is validated through simulation. This section describes simulation environment, fault injection technique and validation of the protocol.

5.1 Environmental Setup

Experiments are performed on simulations with 1000 sensors and 3 gateways uniformly distributed in a 10 × 10 square kilometer area. Each sensor is assumed to have an initial energy of 5 joules. A node is considered non-functional if its energy level reaches 0 joules. The sensor energy consumption model used in our system is discussed in [8] [13].The maximum range of the sensors is set to 0.5 times the maximum distance between two nodes in the system. Initial range of the gateways is considered enough to cover the whole area. It is assumed that the channel is collision free and packets are not dropped in the medium. Sensors are given IDs in random fashion. Sensors are informed about the first TDMA schedules by their respective gateways. Schedule for first inter-gateway communication is decided during bootstrapping. Nodes switch on their transmitter if needed and receiver circuitry only during their allocated slots.

Fault injection is used to test the robustness and behavior of the sensor network. Fault injection allows studying the effectiveness fault detection and recovery capabilities of our system. We use a simulation-based fault injection methodology to inject communication faults in the gateways. We created a fault library of possible link, range and complete failures. We then created a timely ordered failure list using Poisson distribution for occurrence of faults. Faults are picked from the library based on a Uniform distribution for the type of failure and Normal distribution for the location of the fault.

We implement a fault-injector module to trigger the faults as events. The fault-injector keeps a check on the system time and compares it with the timestamp of the next entry in the failure list. Whenever the system time equals a fault event time, the fault-injector selects the fault from the library, reads the fault destination (gateway id) and inserts the fault in the event queue of the gateway. When a gateway encounters the fault in the event queue, it simulates the fault. The consequences of a fault experienced by the system are based on its type.

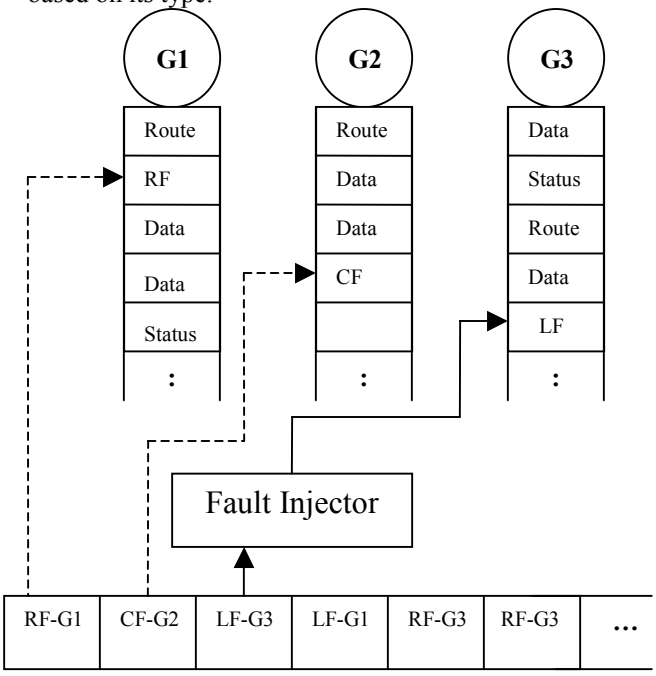

Fig 5. Design of the fault injection environment

Fig 5 describes the design of the fault-injection environment. It shows a link fault being injected by the fault-injector in gateway G3. Previously, gateway G2 stops all operation due to complete failure. Also, gateway G1 is suffering from a range fault. In order to measure the performance of our approach we calculate the coverage of the algorithm. Coverage is the ability of the system to detect and recover from the occurrence of a fault during normal system's operation [17].

Coverage = Probability [system recovers/fault occurs]

We have injected 1000 faults in order to measure the coverage of our algorithm. Since, complete failures are less common compared to other faults, we have inserted them

with lower frequency than range and link failures. All complete failures are detected instantaneously during the status period. We injected faults to decrease the range of gateway G1 by 2% every 15 min of operation. On all occasion our detection mechanism detects the faults and identifies the sensors that have gone out of range from the gateways. Sensors are successfully recovered to other gateways till the range of G1 drops below a threshold and only the sensors very close to G1 are left in the cluster. After every subsequent range fault in G1 coverage decreases because the algorithm fails to find any gateway to accommodate the sensors.

Link failures are injected in gateway G3 to study the impact on number of status messages in the system. Due to link failures status messages are not received by other gateways and forwarding scheme is activated. Total number of message per gateway in the forwarding scheme is N+1, where N is the total number of gateways in the system. Each gateway will transmit one status message, one experience message and forward N-1 status messages. The period of status updates is controlled by MILD algorithm until we inject a burst of faults making the recovery more frequent. The coverage of link failure has been observed to be 100% throughout the simulation until all the incoming/outgoing links from a gateway fails. After all link fails, any fault on G3 are detected as a complete failure by other gateways. Recovery of sensors in the cluster of G3 is only done on the first complete failure.

The results of the fault injection experiments clearly demonstrate that the system is resilient to communication faults and recovers efficiently without re-configurations or manual repairs.

6. Conclusions and future work

High-energy gateway node acts as a centralized manager to handle the sensors and serves as a hop to relay data from sensors to a distant command node. In this paper we have introduced a two phase; detect and recover fault-tolerance approach to recover sensors from the failed gateways without shutting down or re-clustering the system. Gateways can suffer from complete, link or range failures caused due to software or hardware faults. Our approach enables fault-tolerance in the system by performing periodic checks on the status of the gateways. Sensors managed by a faulty gateway are recovered by re-associating them to other clusters based on backup information created during the time of clustering.

Our future plan includes extending the clustering model to allow gateway mobility. Also, we plan to integrate bootstrapping and energy-aware routing to our approach.

References

[1] R. Burne, et. al, "A Self-Organizing, Cooperative UGS Network for Target Tracking," *Proc. of SPIE Conference on Unattended Ground Sensor Tech. and Applications II*, Orlando, April 2000.

[2] "Data sheet for the Acoustic Ballistic Module", SenTech Inc., http://www.sentech-acoustic.com/

[3] W. Heinzelman, et. al, "Energy-Scalable Algorithms and Protocols for Wireless Microsensor Networks," *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP '00),* June 2000.

[4] B. Chen, et al., "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks", *Proc. of MobiCom 2001*, Rome, Italy, July 2001.

[5] S. Singh, M. Woo and C. S. Raghavendra, "Power-Aware Routing in Mobile Ad Hoc Networks", *Proc. of ACM MOBICOM'98*, Dallas, Texas, October 1998

[6] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Scalable coordination in sensor networks. *Proc. of ACM/IEEE MobiCom 1999*, Seattle, Washington, August 1999.

[7] K. Arisha, M. Youssef, M. Younis, "Energy-Aware TDMA-Based MAC for Sensor Networks," *IEEE Workshop on Integrated Management of Power Aware Communications, Computing and Networking (IMPACCT 2002)*, May 2002.

[8] G. Gupta, M. Younis, "Load-Balanced Clustering in Wireless Sensor Networks", Submitted to *the IEEE International conference on communications (ICC 2003),* Anchorage, Alaska, May 2003,

[9] J.M. Kahn, R.H. Katz, K.S.J. Pister, *Next century challenges: Mobile networking for 'smart dust'*, Proc. MOBICOM, Seattle, 1999

[10] Burnstein, A., Bult, K., Chang, D, Chang, F. et al. "Wireless Integrated Microsensors"; *Proceedings Sensors EXPO 1996,* Anaheim, CA., 1996

[11] J. Rabaey, J. Ammer, J.L. da Silva, D. Patel, "PicoRadio: Ad-hoc wireless networking of ubiquitous low-energy sensor/monitor nodes," *IEEE Computer Society Workshop on VLSI 2000*, Orlando, FL, pp. 9--12, April 2000.

[12] A. Cerpa and D. Estrin, "ASCENT: Adaptive Self-Configuring Sensor Networks Topologies," *Proc. INFOCOM 2002*, New York, June 2002

[13] W. Rabiner Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocols for Wireless Microsensor Networks," *Hawaii International Conference on System Sciences (HICSS '00)*, January 2000.

[14] D.J Baker and A. Ephremides, "A Distributed algorithm for Organizing Mobile Radio Telecommunication Networks", in the *Proceedings of the 2nd International Conference in Distributed Computer Systems*, April 1981.

[15] M. Gerla and J.T.C Tsai, "Multicluster, mobile, multimedia radio network," ACM/Baltzer Journal of *Wireless networks*, Vol. 1, No. 3, pp. 255-265, 1995.

[16] A.K. Parekh, "Selecting Routers in Ad-Hoc Wireless Networks", *Proceedings of the SBT/IEEE International Telecommunications Symposium*, August 1994

[17] J. B. Dugan and K. S. Trivedi, "Coverage Modeling for Dependability Analysis of Fault-Tolerant Systems", *IEEE Transactions on Computers, 38 (6), pp.775-87, June 1989*

[18] S. Han, K. G. Shin, and H. A. Rosenberg, "DOCTOR: An Integrated Software Fault Injection Environment for Distributed Real-time Systems," *Proceedings of International Computer Performance and Dependability Symposium, Erlangen, Germany, pp. 204-213, April 1995.*

[19] M.C. Hsueh, T.K Tsai, R.K Iyer, "Fault Injection Techniques and Tools", *Computer, April 1997, pp.75-82*

[20] D. Pradhan, Fault-tolerant computer system design. *Prentice Hall Publisher, Englewood Cliffs, New Jersey, USA, 1996.*