Time constrained fault tolerance and management framework for k-connected distributed wireless sensor networks based on composite event detection

A THESIS SUBMITTED
IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN THE DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

by

Anupam Verma
107CS024

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Orissa, India

May 2011

Supervisor: Prof. P. M. Khilar

National Institute of Technology
Rourkela

CERTIFICATE

This is to certify that the work in the thesis entitled **'Time constrained fault tolerance and management framework for k-connected distributed wireless sensor networks based on composite event detection'** by **Anupam Verma**, **Roll No: 107CS024** is a record of an original research work carried out by him under my supervision and guidance in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **Computer Science Engineering** at **National Institute of Technology, Rourkela.**
To the best of my knowledge the matter embodied in the thesis has not been submitted to any other university/institute for the award of any degree or diploma.

Prof. Pabitra Mohan Khilar
Department of Computer Science and Engineering,
National Institute of Technology,
Rourkela-769008

ACKNOWLEDGEMENT

Words will do no justice to the constant support and encouragement extended by Prof. P. M. Khilar who guided this thesis. If the work has seen light today, it is only because of his motivation during some of my most turbulent times.
Learning is a gradual process of assimilation of learned concepts and cultivating it into the knowledge base. It was an honour to learn from illustrious individuals like Prof. B. Majhi, Prof. A. K. Turuk and Prof. S. K. Rath. Whatever little I could learn from them surely helped me smoothen the learning curve, broaden horizons and sustain interest and passion in my work.
I also express my indebtness to my friends who playfully taught me several lessons of a life, something that is a treasure trove for me. Their indirect help in this thesis is worthy of mention.
I would also like to acknowledge the help of Prof. P.K. Sa without whom this thesis would not look as good as it does today.
Finally, a special thaks to the Almighty for seeing this work through and renewing satisfaction and confidence in me.

Anupam Verma

Abstract

Wireless sensor nodes themselves are exceptionally complex systems where a variety of components interact in a complex way. In enterprise scenarios it becomes highly important to hide the details of the underlying sensor networks from the applications and to guarantee a minimum level of reliability of the system. One of the challenges faced to achieve this level of reliability is to overcome the failures frequently faced by sensor networks due to their tight integration with the environment. Failures can generate false information, which may trigger incorrect business processes, resulting in additional costs. Sensor networks are inherently fault prone due to the shared wireless communication medium. Thus, sensor nodes can lose synchrony and their programs can reach arbitrary states. Since on-site maintenance is not feasible, sensor network applications should be local and communication-efficient self-healing. Also, as per my knowledge, no such general framework exist that addresses all the fault issues one may encounter in a WSN, based on the extensive, exhaustive and comprehensive literature survey in the related areas of research. As one of the main goals of enterprise applications is to reduce the costs of business processes, a complete and more general Fault Tolerance and Management framework for a general WSN, irrespective of the node types and deployment conditions is proposed which would help to mitigate the propagation of failures in a business environment, reduce the installation and maintenance costs and to gain deployment flexibility to allow for unobtrusive installation.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 WSNs: An overview

A wireless sensor network (WSN) is a self-organized system of small, independent, low cost, low powered and wirelessly communicating nodes distributed over a large area with one or possibly more powerful sink nodes gathering readings of sensor nodes and, may handle a variety of sensing, actuating, communicating, signal processing, computation, and communication tasks, deployed in the absence of permanent network infrastructure and in environments with limited or no human accessibility. The sink serves as the gateway between the user application and the sensor network. The WSN nodes have no fixed topology, but they can configure themselves to work in such conditions. In addition, wireless sensor nodes themselves are exceptionally complex systems where a variety of components interact in a complex way. Recent advances in sensor technology and wireless communications have enabled design and development of inexpensive, large sensor networks, which are suitable for different civilian, natural, and military applications, such as health environment monitoring, seismic monitoring, space exploration, structural sensing, habitat monitoring, tele medicine, avionics and battlefields surveillance. With multihop wireless communication, sensor nodes have made it possible to build reactive systems that have the ability to monitor and react to physical events/phenomena. In addition to resource constraints, sensor networks are also failure-prone. Therefore, fault tolerance is as critical as other performance metrics such as energy efficiency, latency and accuracy in supporting distributed sensor applications.

1.2 Sources of faults

At least two components of a sensor node, sensors and actuators, will directly interact with the environment and will be subject to a variety of physical, chemical, and biological forces. Therefore, they will have significantly lower intrinsic reliability than integrated circuits in fully enclosed packaging.

In enterprise scenarios it becomes highly important to hide the details of the underlying sensor networks from the applications and to guarantee a minimum level of

reliability of the system. One of the challenges faced to achieve this level of reliability is to overcome the failures frequently faced by sensor networks due to their tight integration with the environment. Failures can generate false information, which may trigger incorrect business processes, resulting in additional costs. Sensor networks are inherently fault prone due to the shared wireless communication medium: message losses and corruption (due to fading, collision and hidden node effect) are the norm rather than exception. Moreover, node failures (due to crash and energy exhaustion) are the commonplace. They are also prone to failure due to hardware failure, communication link errors, malicious attack, and so on. Thus, sensor nodes can lose synchrony and their programs can reach arbitrary states. Since on-site maintenance is not feasible, sensor network applications should be local and communication-efficient self-healing.

Maintainence of continuous connectivity in a wireless sensor network after it is deployed in a hostile environment is also a major issue. Constrained by the low usertonode ratio, limited energy and bandwidth resources, entities that are usually mobile, networks without fixed infrastructure and frequent failure due to problems of energy, vulnerability to attack etc, a need for wireless sensor networks to be self-organizing and self-configuring so as to improve performance, increase energy efficiency, save resources and reduce data transmission arises. Also, a WSN is prone to several types of faults, such as crash fault, transient fault, byzantine fault etc that affect the normal functioning of the WSN system. Thus, fault tolerance is a major issue confronting the development of highly scalable distributed WSN.

Data delivery in sensor networks is inherently faulty and unpredictable. Failures in wireless sensor networks can occur for various reasons. First, sensor nodes are fragile, and they may fail due to depletion of batteries or destruction by an external event. In addition, nodes may capture and communicate incorrect readings because of environmental influence on their sensing components. Second, as in any ad hoc wireless networks, links are failure-prone, causing network partitions and dynamic changes in network topology. Links may fail when permanently or temporarily blocked by an external object or environmental condition. Packets may be corrupted due to the erroneous nature of communication. In addition, when nodes are embedded or carried by mobile objects, nodes can be taken out of the range of communication. Third, congestion may lead to packet loss. Congestion may occur due to a large number of nodes simultaneous transition from a powersaving state to an active transmission state in response to an event-of-interest [26].

Furthermore, all of the above fault scenarios are worsened by the multihop communication nature of sensor networks. It often takes several hops to deliver data from a node to the sink; therefore, failure of a single node or link may lead to missing reports from the entire region of the sensor network. Additionally, congestion that starts in one local area can propagate all the way to the sink and affect data delivery from other regions of the network.

1.3 The Need for Fault Tolerant Protocols and Design Challenges

Sensor networks share common failure issues (such as link failures and congestion) with traditional distributed wired and wireless networks, as well as introduce new fault sources (such as node failures). Fault tolerant techniques for distributed systems include tools that have become industry standard such as SNMP and TCP/IP, as well as more specialized and/or more efficient methods that have been extensively researched [26]. The faults in sensor networks cannot be approached in the same way as in traditional wired or wireless networks due to the following reasons:

- traditional network protocols are generally not concerned with energy consumption, since wired networks are constantly powered and wireless ad hoc devices can get recharged regularly;
- traditional network protocols aim to achieve point-to-point reliability, whereas wireless sensor networks are concerned with reliable event detection;
- in sensor networks, node failures occur much more frequently than in wired, where servers, routers and client machines are assumed to operate normally most of the time; this implies that closer monitoring of node health without incurring significant overhead is needed;
- traditional wireless network protocols rely on functional MAC layer protocols that avoid packet collisions, hidden terminal problem and channel errors by using physical carrier sense (RTS/CTS) and virtual carrier sense (monitoring the channel).

In wireless sensor networks, MAC layer protocols have to meet other challenges (such as coordinating a nodes sleeping and wake times), and can only mitigate the packet collision problem, not completely solve it. These observations indicate that new fault tolerant protocols are necessary for sensor applications to operate successfully and that these protocols should ensure reliable data delivery while minimizing energy consumption.

1.4 Taxonomy of Fault Tolerant Techniques

Recent research has developed several techniques that deal with different types of faults at different layers of the network stack. To assist in understanding the assumptions, focus, and intuitions behind the design and development of these techniques, we borrow the taxonomy of different fault tolerant techniques used in traditional distributed systems [26]:

- fault prevention: this is to avoid or prevent faults;
- fault detection: this is to use different metrics to collect symptoms of possible faults;

- fault isolation: this is to correlate different types of fault indications (alarms) received from the network, and propose various fault hypotheses;
- fault identification: this is to test each of the proposed hypotheses in order to precisely localize and identify faults;
- fault recovery: this is to treat faults, i.e., reverse their adverse effects.

Fault identification and isolation, sometimes are collectively referred to as fault diagnosis. Note that there do exist some techniques that address a combination of all these aspects. In fact, these techniques operate at different layers of the network protocol stack. Most fault avoidance techniques operate in the network layer, adding redundancy in routing paths; a majority of fault detection and recovery techniques operate at the transport layer; and a few fault recovery techniques perform at the application layer, concealing faults during off-line data processing. Fault tolerance is the ability of a system to deliver a desired level of functionality in the presence of faults [8]. Since the sensor nodes are prone to failure, fault tolerance should be seriously considered in many sensor network applications. Actually, extensive work has been done on fault tolerance and it has been one of the most important topics in WSNs. An early survey work can be found in [36]. However, its coverage is very limited and its references are outdated.

1.5 Representation and Modelling

The power supply on each node is relatively limited, and frequent replacement of the batteries is often not practical due to the large number of the nodes in the network. In order to save energy, nodes only use the short range communications which is proven to be much less energy consuming than the long range. The short range communication between the nodes implies localized interaction in the network. There is a need to model the different components of a sensor network. Sensor networks are often abstracted and mapped into a graph, where each vertex of the graph corresponds to a wireless node and there is an edge corresponding to the communication between two nodes. If the communication between the nodes is bidirectional, the mapped graph of the network will be non-directed. However, if the communication between the nodes is asymmetric, then the mapped graph becomes directed. The communication model between the nodes can be either one-to-one, or one-to- many. In the one-to-one communication model, each node sends and receives messages from only one of the communication edges. In the one-to many communication model, each message sent out by a node can be heard by all of its neighbors. Providing a reasonable and practical model for sensors and actuators is a much more complex task. This is due to the great variety of different sensors, both in terms of their functionality and in terms of their underlying technologies.

Chapter 2

Literature Review

2.1 Fault detection: An Overview

Fault detection is the first phase of fault management, where an unexpected failure should be properly identified by the network system. The existing failure detection approaches in WSNs can be classified into two types: centralized and distributed approach.

2.1.1 Centralized Approach

Centralized approach is a common solution to identify and localize the cause of failures or suspicious nodes in WSNs. Usually, a geographically or logically centralized sensor node (in terms of base station [5, 17, 18], central controller or manager [4], sink [19]) takes responsibility for monitoring and tracing failed or misbehaviour nodes in the network. Most these approaches consider the central node has unlimited resources (e.g. energy) and is able to execute a wide range of fault management maintenance. They also believe the network lifetime can be extended if complex management work and message transmission can be shifted onto the central node. The central node normally adopts an active detection model to retrieve states of the network performance and individual sensor nodes by periodically injecting requests (or queries) into the network. It analyzes these information to identify and localize the failed or suspicious nodes. More specifically, Sympathy [19] uses a message-flooding approach to pool event data and current states (metrics) from sensor nodes. In order to minimize the number of communication messages nodes must send and conserve node energy, a Sympathy node can selectively transmit important events to the Sympathy sink node. While, Jessica Staddon et al.,[18] seek the solution of appending network topology information (i.e. node neighbour list) into nodes routing update messages rather than in a separate approach. Thus, the base station can construct the entire network topology by integrating each piece of network topology information embedded in route update messages. Moreover, some common routing protocols (e.g. SPINs[5]) can also detect failed or misbehaving nodes through routing discovery and update. This typically requires the nodes to send additional messages, and it is consequently very expensive. In [17], the base station uses marked packets (containing geographical information of

source and destination locations etc) to probe sensors. It relies on nodes response to identify and isolate the suspicious nodes on the routing paths when an excessive packet drops or compromised data has been detected. In addition, the central manager in WinMS [4] provides a centralized approach to prevent the potential failure by comparing the current or historical states of sensor nodes against the overall network information models (i.e. topology map, and energy map). As a summary, the centralized approach is efficient and accurate to identify the network faults in certain ways. However, resource-constrained sensor networks can not always afford to periodically collect all the sensor measurements and states in a centralized manner. A distinctive problem of this approach is that the central node easily becomes a single point of data traffic concentration in the network, as it is responsible for all the fault detection and fault management. This subsequently causes a high volume of message traffic and quick energy depletion in certain regions of the network, especially the nodes closer to the base station. They take extra burdens for forwarding the communication messages from other nodes. This approach will become extremely inefficient and expensive in consideration of a large-scale sensor network. In addition, multi-hops communication of this approach will also increase the response delay from the base station to faults occurred in the network. Therefore, we have to seek a localized and more computationally efficient fault detection model.

2.1.2 Distributed Approach

Distributed approach encourages the concept of local decision-making, which evenly distributes fault management into the network. The goal of it is to allow a node to make certain levels of decision before communicating with the central node. It believes the more decision a sensor can make, the less information needs to be delivered to the central node. In the other word, the control centre should not be informed unless there is really a fault occurred in the network. Examples of such development are: node fault self-detection and self-correction on its hardware physical malfunction (i.e. sensor, battery, RF transceiver) [20, 21], failure detection via neighbour coordination [7-10, 22], utilization of WATCHDOG to detect misbehaving neighbour [6], use of group (or cluster) technology to distribute fault detection into the network [2, 23]. Others address the use of decision fusion centre (i.e. several fusion nodes across the network) to make the final decisions on suspicious nodes in the network [11, 12, 14, 16].

Node Self-Detection

S Harte et al., [21] propose a self detection model to monitor the malfunction of the physical components of a sensor node via both hardware and software interface. Self-detection of node failure in [20] is somehow straightforward as the node just observes the binary outputs of its sensors by comparing with the pre-defined fault models. In data dissemination protocols which deliver large segments of data to the entire (or part of the) network, the destination nodes are responsible for detecting the missing packet

or the window of missing packets, and communicating the feedback to the source using NACK messaging such as in PSFQ [28] and GARUDA [29]. In data collection protocols, due to redundancy in sensor nodes and hence the huge amount of reported sensing values, individual packet loss is rarely detected. Instead, a cumulative metric such as packet delivery rate or fault rate is considered [30, 31]. If a certain threshold is exceeded, communication is considered faulty and appropriate recovery actions are taken. In addition to packet loss, other metrics such as interruption, delay or lack of regular network traffic are also considered as symptoms of faults [19, 32]. Alternatively, buffer occupancy level and channel loading conditions [30, 33] are used for fault detection (specifically, congestion). Sensor nodes may also permanently fail. Tools such as ping or traceroute use ICMPmessages to check whether a node is alive or not in wired networks. This approach can also be applied to evaluate the health of sensor nodes. In addition, since sensor nodes are energy-constrained and energy depletion often causes node death, remaining energy level can also be used as a warning of node failure [34, 35].

Neighbour Coordination

Failure detection via neighbor coordination is another example of fault management distribution. Nodes coordinate with their neighbors to detect and identify the network faults (i.e. suspicious node or abnormal sensor readings) before consulting with the central node. For example, in a decentralized fault diagnosis system [22], a sensor node can execute a localized diagnosis algorithm in steps to identify the causes of a fault. In addition, a node can also query diagnostic information from its neighbours (in one-hop communication range). This allows the decentralized diagnostic framework to scale easily to much larger and denser sensor networks if required. Alternatively, suspicious (or failed) nodes can be identified via comparing its sensor readings with neighbors median readings. With this motivation, Min et al., [9] developed a localized algorithm to identify suspicious node whose sensor readings have large difference against the neighbors. Although this algorithm works for large size of sensor networks, the probability of sensor faults needs to be small. If half of the sensor neighbors are faulty and the number of neighbors is even, the algorithm cannot detect the faults as efficient as expected. In addition, this approach also requires each sensor node to be aware of its physical location by equipped with expensive GPS or other GPS-less technology. While, in [10], Jinran et al., improved such kind of approach, which does not require node physical position. This algorithm can still successfully identify suspicious nodes even when half neighbors are faulty. It chooses the GD sensor in the network, and uses its best sensor results to diagnose other sensors status. This information can be further propagated through the entire network to diagnose all other sensors as good or faulty. Chihfan et al., [7, 8] address the accuracy of failure detection via a two-phase neighbour coordination scheme. A sensor node always consults with its neighbours first to verify the diagnosis results before sending out a failure alarm. Similar approach in [6], where a node can listen on its neighbour using WATCHDOG. It can detect failed or misbehaving neighbours if data packets have not

been transmitted properly by the neighbours it is currently routing to. However, this process may be slow, and is error-prone because the constrained sensor nodes cannot be expected to constantly police all their neighbours and consequently may end up routing into a new neighbour that has also failed.

Clustering Approach

Clustering [24] has become an emerging technology for building scalable and energy balanced applications for WSNs. Ann T.Tai et al., [2] derive an efficient failure detection solution using a cluster-based communication hierarchy to achieve scalability, completeness, and accuracy simultaneously. They split the entire network into different clusters and subsequently distribute fault management into each individual region. Intracluster heartbeat diffusion is adopted to identify failed nodes in each cluster. It makes the local failure detection to be aware of the changes of network conditions or the overall objectives. While, Ruiz et al., [23] adopt an event-driven detection via a manager-agent model supported by a management architecture MANNA [3]. In this approach, agents are executed in the cluster-heads with more resources than common nodes. A manager is located externally to the WSN where it has a global vision of the network and can perform complex management tasks and analysis that would not be possible inside the network. Every node checks its energy level and sends a message to the manager or agent whenever there is a state change. The manager then uses these information to build topology map and network energy model for monitoring and detecting the potential failure of the network in future. The scheme has a drawback of providing a false de bugging diagnostic. For instance, common-nodes may be disconnected from its cluster head and so they are not able to receive the GET operation from the manager, or GET and GET-RESPONSE packet may be lost due to noise. These conditions may mislead the manager into making incorrect fault detection. Furthermore, random distribution and limited transmission range capability of common-node and cluster-heads provides no guarantee that every common-node can be connected to a cluster head. In addition, the transmission costs for network state polling has not been considered in this approach.

Distributed Detection

The basic idea of Distributed Detection is to have each node make a decision on faults (typically binary data of abnormal sensor reading). This approach is especially energy-efficient and ideal for datacentric sensor applications. However, there remain various research challenges in order to achieve a better balance between fault detection accuracy and the energy usage of the network. Usually, the efficiency of such failure detection schemes is counted in terms of node communication costs, precision, detection accuracy and the number of faulty sensor nodes tolerable in the network. One of the techniques suggested is fusion sensor coordination. In Clouqueurs work [15], fusion sensors (in terms of manager nodes) coordinate with each other to guarantee that they obtain the same global information about the network before making a decision, as faulty nodes may send them inconsistent information. In addition, Wang

et al. [16] also consider a fault-tolerant solution to reduce the overall computation time and memory requirements at the fusion sensors. This approach adopts cluster technology for data aggregation and lessening of redundant data. In [15], Thomas et al., seek efficient algorithms for collaborative failure target detection that are efficient in terms of communication cost, precision, accuracy, and number of faulty sensors tolerable in the network. Fusion sensors (in terms of manager nodes) coordinate with each others in order to guarantee that all manager nodes obtain the same global information of the network, as faulty nodes may send inconsistent information to different manager nodes. While, Tsang-Yi et al., [16] look into the design of a fault-tolerant fusion rule for wireless sensor networks.

2.2 Fault Diagnosis: An Overview

Fault diagnosis is a stage that the causes of detected fault can be properly identified and distinguished from the other irrelevant or spurious alarms. The accuracy and correctness of detected fault have already been partly reviewed and achieved in fault detection phase as in [2, 8, 12, 15]. However, there is still no comprehensive model or description of faults in sensor networks to support the network system for accurate fault diagnosis. Most existing approaches address the fault models only on the individual node level (including its hardware components malfunction). In particular, both [10, 20] assume the system software (including sensor application software) are already fault tolerant. They focus on hardware level faults of a sensor node, especially on sensor and actuator which are most prone to malfunctioning. Farinaz et al., [20] adopt two fault models. The first one is related to sensors that produce binary outputs. The second fault model is related to the sensors with continuous (analog) or multilevel digital outputs. In [15], Thomas et al., only consider faulty nodes are due to harsh environmental conditions. In their work, faulty nodes are assumed to send inconsistent and arbitary values to other nodes during information sharing phase. While, Min Ding et al., [9] model the event (or abnormal behaviour of a sensor node) by real numbers such as sensor readings instead of 0/1 decision model. As a result, this algorithm is generic enough as long as the thresholds and real number of events can be specified by fault tolerance requirements from various sensor applications.

With detected alarms, fault isolation and identification processes will diagnose and determine the real causes.When the sink does not hear from a particular part of the routing tree, it is unknown whether it is due to failure of a key routing node, or failure of all nodes in a region. A fault tracing protocol has been proposed [19] to differentiate between these two cases. This is achieved in two steps. Each individual node first piggybacks its neighbor nodes IDs to the sink along with its own readings so that the sink can have a complete network topology. Failed nodes can then be traced by using a divide-and-conquer strategy based on adaptive route update messages. The sink broadcasts a route update to determine whether the silent nodes are dead. This approach does not perform well in a large scale sensor network: if there are constant failures, the sink would be frequently broadcasting routing updates, which would cause

significant overhead. It is desirable that nodes can make some local decisions about fault severity. Sympathy [32] considers three possible sources of failures for a node: self, path and sink. Sympathy monitors regular network traffic which is assumed to be frequently generated by each healthy node: sensor readings, synchronization beacons, routing updates, etc. Sympathy treats absence of monitored traffic as an indication of faults. It uses metrics traffic generated at the nodes to localize the failure. These metrics include connectivity metrics (e.g., routing table, neighbor list), flow metrics (e.g., packets transmitted and received per node and per sink), and node metrics (e.g., uptime). The measurements expire if they are not updated for a certain period of time. Sympathy determines whether the cause of failure is in node health, bad connectivity/connection, or at the sink by using an empirical decision tree.

2.3 Routing Protocols in WSNs: An Overview

In general, routing in WSN can be categorized based on

- Network Structure
- Protocol Operation
- How the source finds a route to the destination
- Extent to which the mobility is allowed

The above classification and this section is based on [38].

2.3.1 Based on network structure

Routing in WSNs can be divided into flat-based routing, hierarchical-based routing, and location-based routing depending on the network structure.

Flat-based routing

In flat-based routing, all nodes are typically assigned equal roles or functionality and sensor nodes collaborate together to perform the sensing task. Due to the large number of such nodes, it is not feasible to assign a global identifer to each node. This consideration has led to data centric routing, where the BS sends queries to certain regions and waits for data from the sensors located in the selected regions. Since data is being requested through queries, attribute-based naming is necessary to specify the properties of data. Eg.: SPIN, Directed Diffusion, Rumour Routing, COUGAR, ACQUIRE, EAD, Information Directed Routing, Gradient Based Routing, Quorum Based Information dissemination Routing, Home Agent Based Information dissemination Routing and Energy Aware Routing are all data centric routing protocols. IDSQ, CADR, MCFA and routing protocols that use random walks are also flat-based routing protocols.

Hierarchical-based routing

In hierarchical-based routing, however, nodes will play different roles in the network. Hierarchical or cluster-based routing, originally proposed in wireline networks, are well-known techniques with special advantages related to scalability and efficient communication. As such, the concept of hierarchical routing is also utilized to perform energy-effcient routing in WSNs. In a hierarchical architecture, higher energy nodes can be used to process and send the information while low energy nodes can be used to perform the sensing in the proximity of the target. This means that creation of clusters and assigning special tasks to cluster heads can greatly contribute to overall system scalability, lifetime, and energy efficiency. Hierarchical routing is an efficient way to lower energy consumption within a cluster and by performing data aggregation and fusion in order to decrease the number of transmitted messages to the BS. Hierarchical routing is mainly two-layer routing where one layer is used to select clusterheads and the other layer is used for routing. However, most techniques in this category are not about routing, rather on "who and when to send or process/aggregate" the information, channel allocation etc., which can be orthogonal to the multihop routing function. Eg.: LEACH, PEGASIS, TEEN, APTEEN, MECN, SMECN, SOP, Sensor Aggregates Routing (DAM, EBAM, EMCAM), HPAR, TTDD and Virtual Grid Architecture Routing (VGA, further divided into ILP and CBAH) are all hierarchical-based routing protocols.

Location-based routing

In location-based routing, sensor nodes' positions are exploited to route data in the network. In this kind of routing, sensor nodes are addressed by means of their locations. The distance between neighboring nodes can be estimated on the basis of incoming signal strengths. Relative coordinates of neighboring nodes can be obtained by exchanging such information between neighbors. Alternatively, the location of nodes may be available directly by communicating with a satellite, using GPS (Global Positioning System), if nodes are equipped with a small low power GPS receiver. To save energy, some location based schemes demand that nodes should go to sleep if there is no activity. More energy savings can be obtained by having as many sleeping nodes in the network as possible. Eg.: GAF, GEAR, MFR, DIR, GEDIR, GOAFR and SPAN are all location-based routing protocols.

2.3.2 Based on protocol operation

The routing protocols for WSNs can be classified into multipath-based, query-based, negotiation-based, QoS-based, or coherent-based routing techniques depending on the protocol operation.

Multipath-based

Here, the routing protocols use multiple paths rather than a single path in order to enhance the network performance. The fault tolerance (resilience) of a protocol

is measured by the likelihood that an alternate path exists between a source and a destination when the primary path fails. This can be increased by maintaining multiple paths between the source and the destination at the expense of an increased energy consumption and trac generation. These alternate paths are kept alive by sending periodic messages. Hence, network reliability can be increased at the expense of increased overhead of maintaining the alternate paths. Eg.: HPAR, Directed Diffusion, Ganesan et. al. model, Maximum lifetime routing, Dulmann et. al. model, Rahul-Rabacy model etc..

Query-based

In this kind of routing, the destination nodes propagate a query for data (sensing task) from a node through the network and a node having this data sends the data which matches the query back to the node, which initiates the query. Usually these queries are described in natural language, or in high-level query languages. For example, client C1 may submit a query to node N1 and ask: Are there moving vehicles in battle space region 1?. All the nodes have tables consisting of the sensing tasks queries that they receive and send data which matches these tasks when they receive it. Eg.: Directed Diffusion, Rumour Routing,GPS-less low cost outdoor localization model, SPIN, GBR, EAR, COUGAR and ACQUIRE

Negotiation-based

These protocols use high level data descriptors in order to eliminate redundant data transmissions through negotiation. Communication decisions are also taken based on the resources that are available to them. Eg.: SPIN, Balakrishna's model, VGA, SPAN and SAR

QoS-based

In QoS-based routing protocols, the network has to balance between energy consumption and data quality. In particular, the network has to satisfy certain QoS metrics, e.g., delay, energy, bandwidth, etc. when delivering data to the BS. Eg.: SAR, SPEED

Coherent-based and Non coherent-based

Data processing is a major component in the operation of wireless sensor networks. Hence, routing techniques employ different data processing techniques. In general, sensor nodes will cooperate with each other in processing different data flooded in the network area. Two examples of data processing techniques proposed in WSNs are coherent and non-coherent data processing-based routing. In non-coherent data processing routing, nodes will locally process the raw data before being sent to other nodes for further processing. The nodes that perform further processing are called the aggregators. In coherent routing, the data is forwarded to aggregators after minimum processing. The minimum processing typically includes tasks like time stamping, duplicate suppression, etc. To perform energy-effcient routing, coherent processing

is normally selected. Non-coherent functions have fairly low data traffic loading. On the other hand, since coherent processing generates long data streams, energy effciency must be achieved by path optimality. In non-coherent processing, data processing incurs three phases: (1) Target detection, data collection, and preprocessing (2) Membership declaration, and (3) Central node election. During phase 1, a target is detected, its data collected and preprocessed. When a node decides to participate in a cooperative function, it will enter phase 2 and declare this intention to all neighbors. This should be done as soon as possible so that each sensor has a local understanding of the network topology. Phase 3 is the election of the central node. Since the central node is selected to perform more sophisticated information processing, it must have suffcient energy reserves and computational capability. Eg.: SWE and MWE

2.3.3 Based on how the source finds a route to the destination

Routing protocols can be classified into three categories, namely, proactive, reactive, and hybrid protocols depending on how the source finds a route to the destination. In proactive protocols, all routes are computed before they are really needed, while in reactive protocols, routes are computed on demand. Hybrid protocols use a combination of these two ideas. When sensor nodes are static, it is preferable to have table driven routing protocols rather than using reactive protocols. A significant amount of energy is used in route discovery and setup of reactive protocols.

2.3.4 Extent to which the mobility is allowed

Routing Protocols may allow limited mobility, full mobility and mobility with a fixed base station.

Chapter 3

Proposed Work

3.1 Assumptions and Scope

A fault tolerance model for a WSN is envisioned based on a few basic assumptions:

- The capabilities of the sensors can be heterogeneous, that is different sensors can have different sensing capabilities.
- Sensor nodes are immobile in the system, but the mobility of the user node is allowed.
- Wireless broadcast is used for communication

This is based on [37], but it addresses the issue of k coverage, whereas the model I intend to design would hold for both k connected as well as k covered WSNs.

Also, following classes of faults would be addressed:

- faults arising due to low energy or energy exhaustion
- faults arising due to node failures
- faults arising due to hardware problems or faulty hardware components
- faults arising due to misbehaviour of nodes, such as faulty reporting of data, crashing of nodes, malicious attack etc.
- faults arising due to collision and hidden node effect
- faults arising due to message losses and corruption
- faults arising due to communication link errors
- transient fault, byzantine fault
- faults arising due to congestion

The above proposed model will address all the above faults based on the multihop communication nature of sensor networks. It bulds extensively on the basics and fundamentals developed over the decade in the areas of computer communication and fault tolerant techniques in MANETs, WSNs and distributed systems. The model involves development of certain mathematical models based on the real time reporting of faults, which is non deterministic (i.e., its time of occurence is not known) and non probabilistic (i.e., where probabilities of occurence of events may not be the best metric to run heuristics to detect faults). The framework is quasi distributed and quasi decentralized in nature, something that has not been addressed in any of the papers surveyed so far. It intends to combine the merits of centralized and distributed approaches into an altogether different model which would use abstractions at every layer of fault propagation for an effective fault diagnosis. The proposed framework would also use polling to detect certain types of faults. However, considering the deadlines and resource availability, fault prevention and fault recovery shall fall beyond the scope of the present research work.

3.2 System Model

The system consists of a set of energy constrained sensor nodes, wirelessly communicating with one another, via radio transceivers. A wireless link is established between two nodes only if they are in range of each other. Communication between nodes is over a single shared channel. The nodes can be homogeneous or heterogeneous in nature with respect to the sensing and communication ranges, memory capacity, processing power and battery power. We assume that all transmissions from any node are omnidirectional i. e., any message sent by u can be received by any node in its neighborhood, i.e. within its transmitting range. The nodes are immobile at the time of ID assignment. The nodes can detect channel congestion and packet collisions. Also, the node deployment can be either deterministic or self-organizing. The area in which nodes are deployed is divided into several regions, each headed by a regional head which is not one of the nodes. The concept of regional head is analogous to that of a sub-station or base station equipped with better, long haul communication facilities and unlimited power supply. The regional heads are established first before the nodes are deployed. They can sense and communicate directly with all the nodes that fall within their region. The nodes can be thought to contain a special, tamper proof chip analogous to SIM card, which help them to get sensed by other nodes and regional head/s. The chip also helps the nodes to distinguish valid nodes from invalid and rogue or malicious ones. It is also assumed that the regional heads can get the information about the communication and sensing ranges of the nodes using the central database of chip information. The regional heads have their geographical coordinates uniquely identified and are responsible for unique ID assignment and ID pool replenishment for all the nodes in their region. They do the fault and failure diagnosis in consultation with the participating nodes and are also responsible for clock synchronization. All the regional heads report to the central head at regular

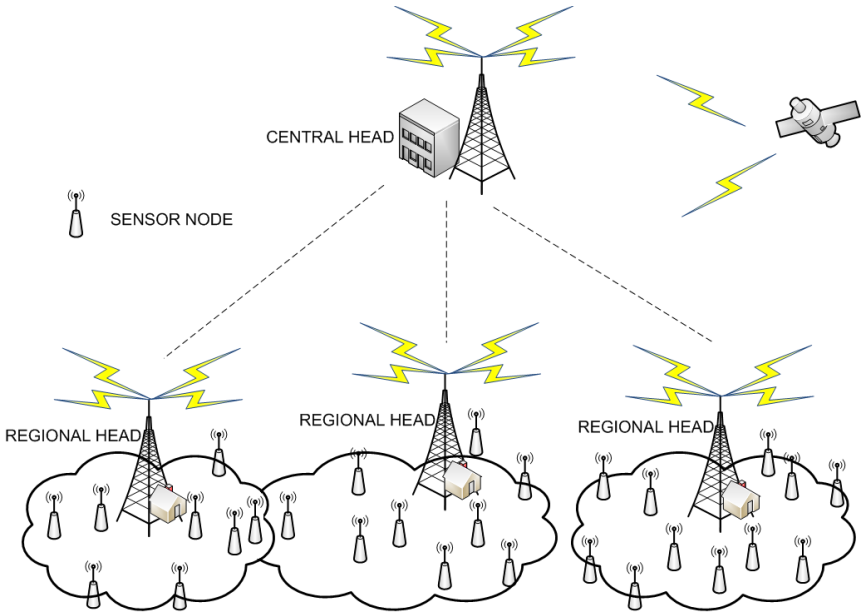

Figure 3.1: System Model

intervals of time. It is they who would be probed for information by the users either directly or indirectly through the central head. They perform the bulk of computation tasks in the network. The connectivity and coverage in the network are regularly monitored by them. For the sake of simplicity, we also assume that no overlap occurs between the regions though; we feel that the model can be extended with suitable modifications to such cases. In a nutshell, the network follows a hierarchical structure with the central head forming the root node, with the regional heads as their children. Sensor nodes can be visualized as the children of the regional heads.

3.3 Preliminaries

In this section in order to make the thesis self-contained, the relevant key facts and assumptions are outlined.
Sometimes, the situation may demand nodes to report the readings within a certain time span. With reference to our work, we define a term deadline, which is the maximum time limit within which the readings must be reported. If the value of deadline is low, it may trigger an abnormally large number of bits under transmission. To measure this, we define another term called load per sensor, which is the number of bits being processed or transmitted by a sensor per unit time. The basic aim of our model is to maximize the network lifetime, which we define in terms of both connectivity and coverage. Sensing coverage characterizes the monitoring quality provided by a sensor network on a designated region and reflects how well a sensor network is monitored or tracked by sensor. For the network lifetime based on sensor coverage, we follow the most common definition, which uses 1-coverage to define the lifetime as the time that the region of interest is completely within the sensing range of at least one sensor node -the region is covered by at least one node. Connectivity

is the ability of any active node to communicate directly or indirectly with any other active node. For the network lifetime based on connectivity, we define it as the time that the data related to the region of interest can be successfully sent back to the regional head. For the sake of simplicity, we consider that only a finite set of target points inside an area has to be covered (i. e., target coverage). The model can be extended to include area or volume coverage where the region of interest can be a two-dimensional area or a three-dimensional volume, where each point inside the area or volume has to be covered.
For this thesis, N represents the set of nodes, R is the set of regional heads, C is the set of central heads, $\eta(u)$ is the neighborhood of the node u and λ_u and ϕ_u are the latitude and longitude values of the node u respectively.

3.4 ID Determination

One of the most important issues in fault diagnosis is the unique identification of nodes and the resilience of such a scheme under faults and varying topology of a WSN. A geographical identification and routing scheme based on the actual coordinates of a node, and a subsequent assignment of IDs based on the hierarchy is envisioned. However, storing, communicating and processing such IDs derived from the geographical coordinates may not only be inefficient but also highly energy consuming. Also, maintaining such large routing tables would be highly inefficient. These shortcomings can be easily avoided based on a simple observation that if the relative coordinates are known, the packet size can be optimized. Also, if maximum and minimum values of the coordinates are known, the difference between them can be used to save the memory space, minimize the transmission overhead and improve precision. The central head and regional heads are first established before the nodes are deployed. Their geographical coordinates and unique IDs are pre-determined. The number of nodes each region may hold may be deterministic or random. Algorithm-1 can be used for both the cases as the task of managing all the activities within the region lies with the regional head. For latitudes, we use $'+'$ sign to indicate NORTH and $'-'$ sign to indicate SOUTH. For longitudes, we use $'+'$ sign to indicate EAST and $'-'$ sign to indicate WEST. As the nodes come up, the regional head sends a hello packet which has key information such as its ID and the node's relative coordinates with respect to it (say, λ' and ϕ'). These can be easily determined based on the orientation of the antenna at the regional head end, relative to the horizontal and vertical planes. The packet also has ID the node can use and the sink node bit which, if set, indicates the node has been selected to serve as a sink node. The regional head decides the sink nodes based on the network coverage, proximity, hardware resources etc. If the bit is not set, the head also includes ID of the nearest alive sink node. Based on the relative signal strength, the nodes compute the distance. The next step is ID confirmation. The nodes broadcast their IDs to their immediate neighbors (i.e., within one hop neighborhood) who route the packets to the regional head in a multi hop fashion. Note that a node may be reassigned ID if it has just recovered from a fault. In that

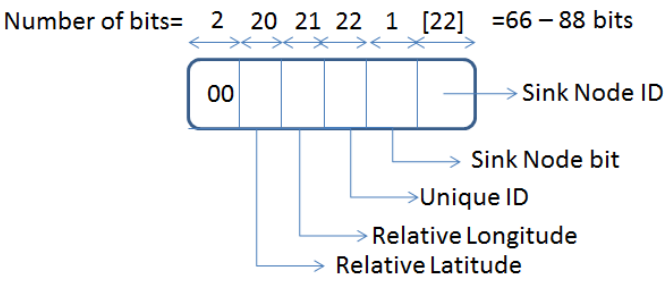

Figure 3.2: Hello Packet

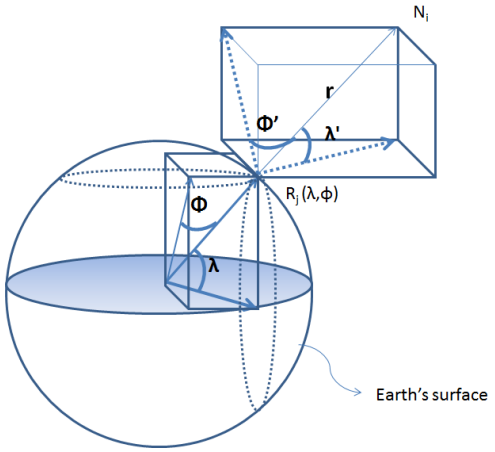

Figure 3.3: Coordinate System of a node relative to the regional head

case, the old ID bit is set, which indicates that the next few bits represent old ID. Now, the node need not send chip ID. So, the chip ID bit is set to 0, and the packet is terminated. Also, if a node has been assigned ID for the first time, the old ID bit is set to 0. In such a case, the next field will not be the old ID field but the chip ID bit field which is set to 1, indicating that the next few bits represent chip ID. Thus, within 2 message transmissions:
a) the nodes come to know about their coordinates and ID and
b) IDs and coordinates of their immediate neighbors.
If the nodes do not respond back, the regional head probes the nodes and starts a timer. If the timer expires, the ID is included into the list of available IDs for future use. After the complete table is built up, the nodes check if there are any sink nodes in their one hop neighborhood. If not, they simply forward readings to any of their alive neighbors who route it to the nearest sink node. The Hello packets are bulky, but the burden of transmitting them lies with the regional head and they are processed only once by the nodes during their course of normal operation, till a fault occurs. The ID consists of three fields: central head field, regional head field and the node ID field. IDs of the form $a.0.0$ represent the central head, $a.b.0$ represent the ID of the regional head and IDs of the form $a.b.c.$, where represent the nodes. Note that $a, b, c \neq 0$. For this paper, $'a'$ is of 2 bits, $'b'$ is of 4 bits and $'c'$ is of 16 bits. This implies that the network has 3 central heads, 15 regional heads and 216-1 nodes per regional head.

Input : 1. A set of nodes, N
2. A set of regional heads, R

Output: 1. Geographical Coordinates (λ'_{N_i} and ϕ'_{N_i}) of $N_i \in N$.
2. IDs of nodes, I
3. Area under regional head to which each node belong, $A = (N, R)$

foreach *node* $N_i \in N$ **do**
  if $\lambda'_{N_i} == \emptyset$ *and* $\phi'_{N_i} == \emptyset$ *or* $I_{N_i} == \emptyset$ **then**
    `wait for hello packet from the regional head`;
    //if the regional head, say R_k *responds*
    //N_i *receives hello packet and maintains information in a table.*
    $\texttt{Table}_{N_i}(R_k) \leftarrow$ `HelloPacket`$(\lambda'_{R_k}, \phi'_{R_k}, I_{R_k \rightarrow N_i}, SinkNodeBit, SinkNodeID)$;
    //where, $I_{R_k \rightarrow N_i}$ *is the ID assigned by* R_k *to* N_i
    //N_i *calculates its distance and stores it in the table*
    $\texttt{Table}_{N_i}(R_k) \leftarrow$ `Distance`(N_i, R_k);
    // Node broadcasts ID confirmation packet to its neighbours
    //indicating its intent. Neighbours can use this step to exchange information.
    foreach *node* $N_j \in \eta(N_i)$ **do**
      //Nodes store information about N_i in a table $\texttt{Table}_{N_j}(N_i) \leftarrow$ `IDConfirm`$(I_{R_k \rightarrow N_i}, \texttt{Distance}(N_i, R_k), \texttt{SinkNodeBit}, \texttt{SinkNodeID})$;
      $\texttt{Table}_{N_i}(N_j) \leftarrow$ `IDConfirm`$(I_{R_t \rightarrow N_j}, \texttt{Distance}(N_j, R_t), \texttt{SinkNodeBit}, \texttt{SinkNodeID})$;
      // where, R_t *is the head of the region to which* N_j *belongs*
    end
  end
end

Algorithm 1: Node Coordinate and ID determination

3.5 Reporting of Readings

After the IDs are confirmed by the nodes, the regional head broadcasts to all the nodes within its region the deadline i.e., the time interval, δ within which they must report their readings to the nearest sink node, the tolerance limit, τ and the maximum time, T for which the node may not respond if the readings are within the tolerance limit. If a node records a reading which deviates from the originally reported reading by τ, it sends a triggered update to the nearest sink node. The sink node sends it to the regional head, which then decides whether it is to be treated as faulty or not based on the data available. If the reading is within the tolerance limit, it is not reported for a time period not exceeding T. This helps to save power and improves the network lifetime. The packet structure allows for the flexibility of the way nodes wish to report the readings. They may either report the readings as it is or report them as deviation, whichever takes less number of bits. This is essentially done to reduce the transmission and processing overhead which may occur if T and δ are small. As a summary, the nodes must report their read-

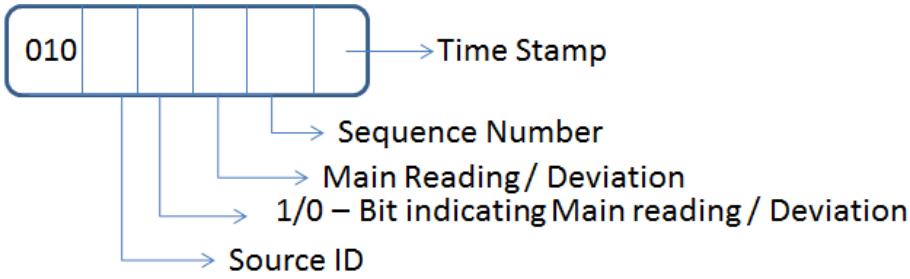

Figure 3.4: Packet for reporting of readings

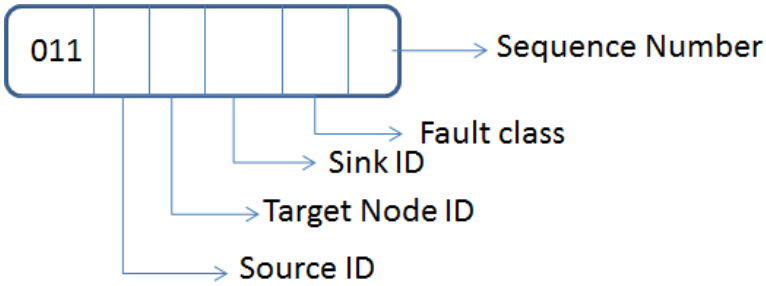

Figure 3.5: Fault Identification Packet

ings when they do so for the first time or when a large deviation from the original reading is reported or when no reading has been reported for a period T.

Input : 1. A set of nodes, G
2. A set of regional heads, R
3. IDs of nodes, I for each $N_i \in N$

Output: Set of Readings, S

```
1 foreach node N_i ∈ N do
2     if no reading has been reported for time ≥ δ or deviation ≥ threshold then
3         // Say, the sink node for N_i region is J_{N_i}, J_{N_i} ∈ N
4         J_{N_i} ← TransmitReading (S_{N_i}, timestamp, SinkNodeID);
5     end
6 end
```

Algorithm 2: Transmission of readings

3.6 Fault and Failure Diagnosis

Before addressing fault and failure diagnosis mechanisms, it is important to point out the difference between faults, errors, and failures.
1. A fault is any kind of defect that leads to an error.
2. An error corresponds to an incorrect (undefined) system state. Such a state may lead to a failure.
3. A failure is the (observable) manifestation of an error, which occurs when the system deviates from its specification and cannot deliver its intended functionality.

3.6.1 Crash or Omission Faults

Crash faults may occur if the nodes get damaged during deployment or during their course of normal operation. A failure by omission is determined by a service sporadically not responding to requests. Such faults can be detected if they do not respond to the repeated requests by other nodes, sink node or regional head. An upper limit, f on the number of requests to which a node does not respond before being classified as crashed. A threshold limit on the expiry timer beyond which the request would be categorized as failed can also be used.

Input : 1. A set of nodes, N
2. A set of regional heads, R
3. IDs of nodes, I for each $N_i \in N$

Output: A set of nodes, N_C experiencing crash faults, if any.

```
1  foreach node N_i ∈ N do
2      foreach node N_j ∈ η(N) do
3          while f_{N_j} ≤ f do
4              Send a packet and start the timer;
5              if no response from N_j and timer ≥ thresholdlimit then
6                  Increment f_{N_j};
7              end
8              else if response from f_{N_j} and timer < thresholdlimit then
9                  Decrement f_{N_j};
10                 break;
11             end
12         end
13         if f_{N_j} ≥ f then
14             Add N_j to N_C;
15             // Inform the sink node
16             SinkNode ← FaultIdentificationPacket(FaultClass = CrashFault);
17         end
18     end
19 end
```

Algorithm 3: Crash fault detection

3.6.2 Interruption, delay or lack of regular network traffic

In addition to packet loss, other metrics such as interruption, delay or lack of regular network traffic are also considered as symptoms of faults. We use a metric called load per sensor on which lower and upper bounds are put. If the load is within these bounds, the node is said to experience normal traffic over its links with other nodes. If the load crosses the upper limit, it goes into sensing state and switches off its transmitters to save energy, and comes into active state only when the need arises for it to transmit readings or fault information. If the load is below the threshold limit, the node starts a timer and monitors the traffic. If the timer crosses the threshold, it probes its neighbors to identify potential faults and report the same to the sink node.

If the node experiences a high rate of traffic interruption, it informs its neighbors and sink node about the same. Note that these limits are hardcoded into the nodes, and may be considered to be same for all the nodes, homogeneous or heterogeneous.

Input : 1. A set of nodes, N
2. A set of regional heads, R
3. IDs of nodes, I for each $N_i \in N$

Output: Identification of faults based on network traffic

foreach *node* $N_i \in N$ **do**
  // Traffic is measured in terms of load per sensor, i.e.,
  //$traffic = f(loadpersensor)$
  if $LowerBound \leq traffic \leq UpperBound$ **then**
    `Continue with the normal operation;`
  end
  else if $traffic < LowerBound$ **then**
    `Start fault diagnosis for the neighbors and itself;`
    `Inform the sink node in case a fault or failure is detected;`
  end
end
else if $traffic > UpperBound$ **then**
  `Switch off the transmitter;`
  `Perform sensing activities only;`
  `Wait for a random period of time not exceeding` T `before going to the acive state;`
end

Algorithm 4: Fault detection based on traffic conditions

Chapter 4

Simulation Results and Conclusion

4.1 Simulation Results

A square grid of 21 X 21 was selected and nodes were randomly deployed. Their communication and sensing ranges were randomly varied to simulate hetrogeneous nodes. Random numbers were generated to simulate faults and asynchronous wakeups. The implentation was done in C and the plotting is done in MATLAB and Microsoft Excel. The proximity to regional heads is calculated using relative signal strength which is directly proportional to the euclidean distance. In case of a tie, the node decides its allegiance to the regional head by looking at its neighbourhood. The results obtained are summarized in figures 4.1 through 4.7.

4.2 Analysis

The figures show that for a given network deployment, load on nodes gets distributed with respect to their sensing and communication capabilities. The linear behaviour for number of message transmissions detecting crash faults looks promising. It can also be seen that though the number of message transmissions shows non linear behaviour,

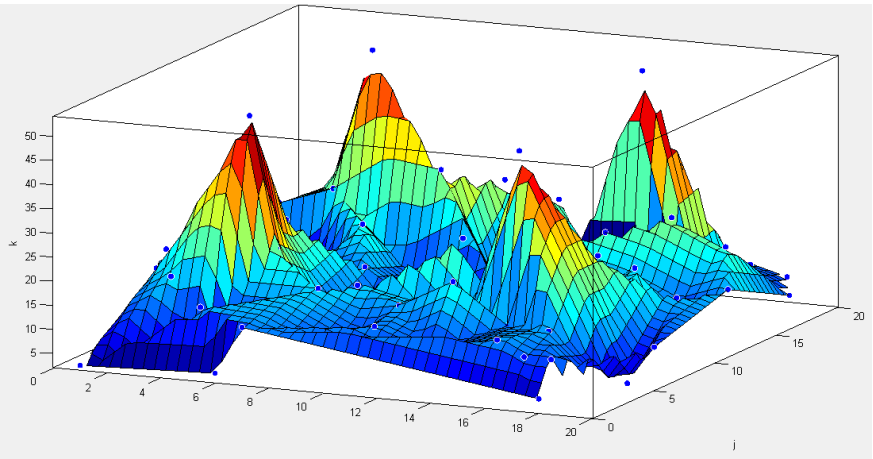

Figure 4.1: Number of message transmissions for ID determination

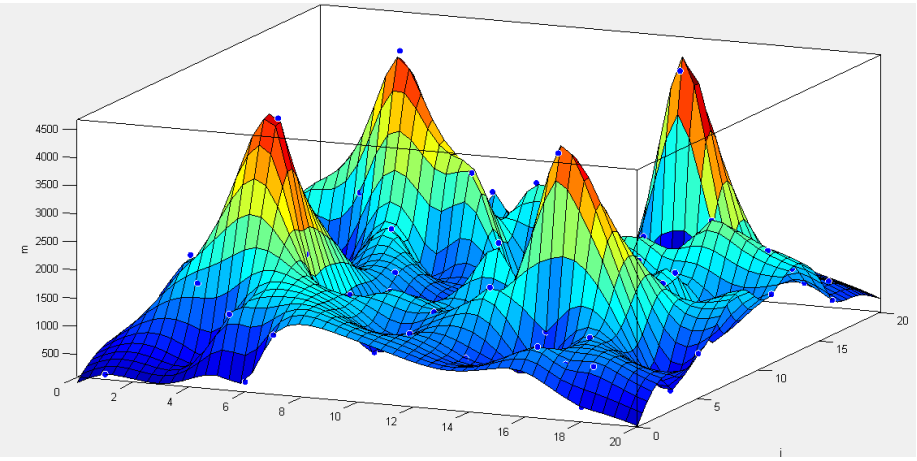

Figure 4.2: Number of bits under transmission for ID determination at a given instant

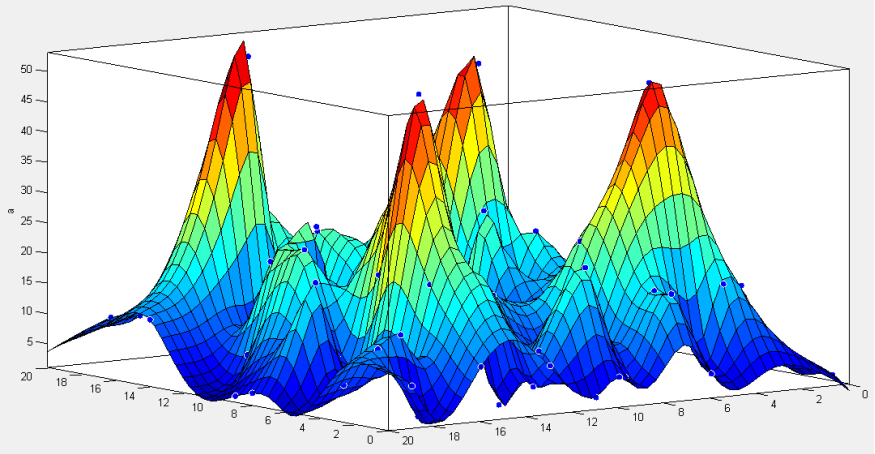

Figure 4.3: Number of message transmissions for reporting of readings at a given instant

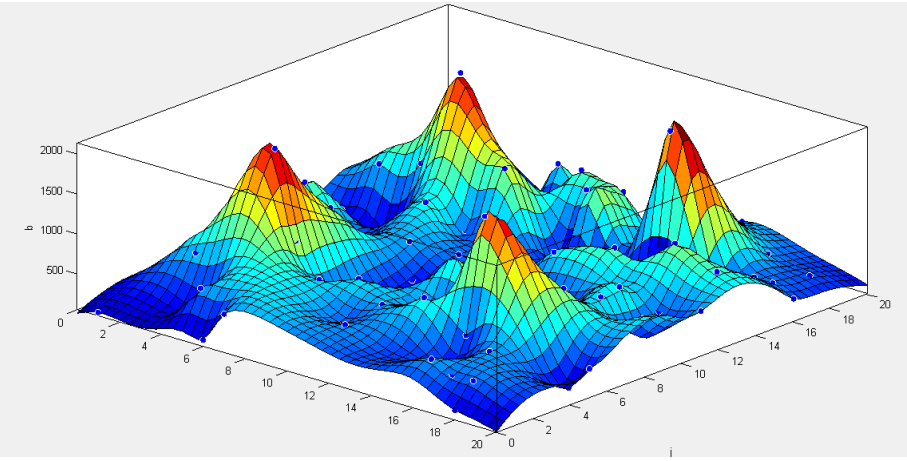

Figure 4.4: Number of bits under transmission for reporting of readings at a given instant

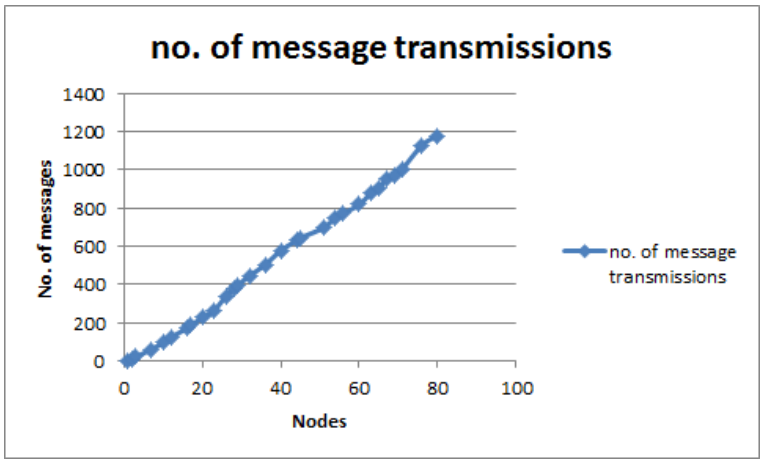

Figure 4.5: Number of message transmissions required to detect crash faults

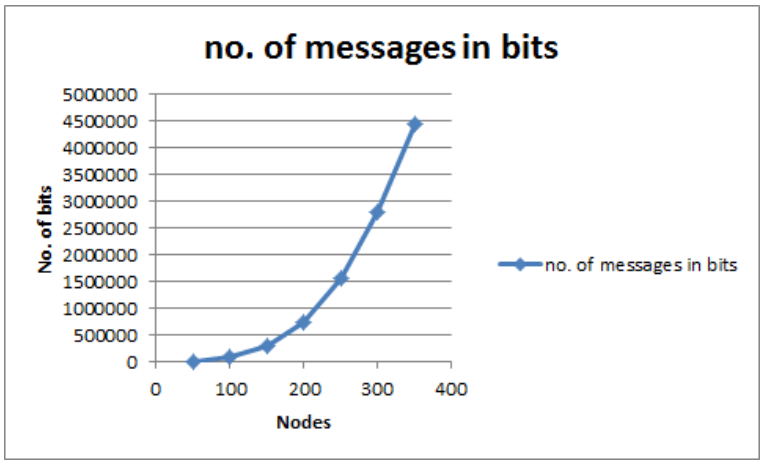

Figure 4.6: Number of bits under transmission for detecting crash faults

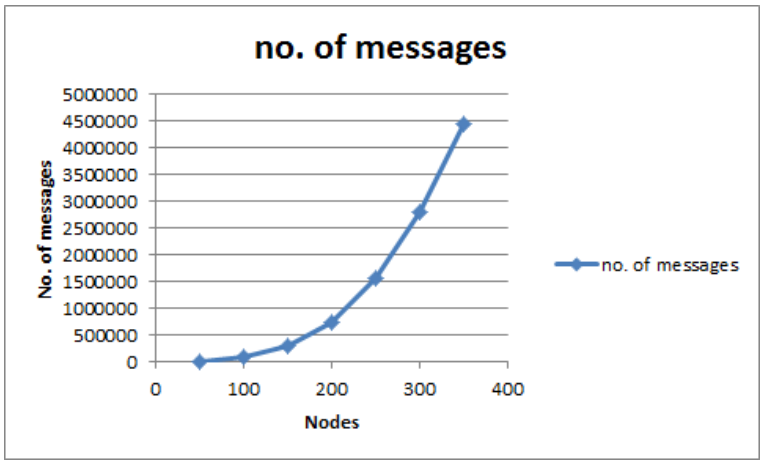

Figure 4.7: Number of message transmissions with rising number of nodes

their values, however, are lower.

4.3 Conclusion

As a part of the implementation, the existing protocols for wired networks were studied and their performance was compared based on a few metrics. This was essential in understanding and appreciating the core concepts of network routing protocols as well as the current state of research, which has helped to address a certain kind of faults. Unique ID assignment, efficient detection of failed nodes, routing and fault and failure diagnosis have been independently addressed by the researchers, and a very few papers exist that incorporate more than one issue. As per the current state of work, the payload has been designed which will be used to address all of them. The payloads were carefully designed keeping in mind the limited resources and the frequencies of a particular task and specific transmissions that plague the existing WSN nodes. Also, the entire framework was designed on one major basic assumption that all the nodes have atleast two neighbours so that the coordinates can be accurately determined upto a desired degree of precision. Also, the framework should be highly scalable, flexible and resilient to faults that affect the network lifetime. Addressing different types of faults, routing under varying topology and efficient reporting of readings so as to reduce the transmission and proceeing overhead is a formidable challenge on which the work is still being done.

Bibliography

[1] Z. Ying and X. Debao, "Mobile agent-based policy management for wireless sensor networks," *Wireless Communications, Networking and Mobile Computing, IEEE*,vol. 2, pp. 1207 – 1210, sep. 2005.

[2] A. Tai, K. Tso, and W. Sanders, "Cluster-based failure detection service for large-scale ad hoc wireless network applications," *International Conference on Dependable Systems and Networks, IEEE*, pp. 805 – 814, jun. 2004.

[3] L. Ruiz, J. Nogueira, and A. Loureiro, "Manna: a management architecture for wireless sensor networks," *Communications Magazine, IEEE*, vol. 41, pp. 116 – 125, feb. 2003.

[4] W. L. Lee, A. Datta, and R. Cardell-oliver, "Winms: Wireless sensor network-management system, an adaptive policy-based management for wireless sensor networks," School of Computer Science and Software engineering, University of western Australia,2006.

[5] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "Spins: security protocols for sensor networks," *Wirel. Netw.*, vol. 8, no. 5, pp. 521–534, 2002.

[6] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, (New York, NY, USA), pp. 255–265, ACM, 2000.

[7] C.-f. Hsin and M. Liu, "A distributed monitoring mechanism for wireless sensor networks," in *WiSE '02: Proceedings of the 1st ACM workshop on Wireless security*, (New York, NY, USA), pp. 57–66, ACM, 2002.

[8] C. Hsin and M. Liu, "Self-monitoring of wireless sensor networks," *Comput. Commun.*, vol. 29, no. 4, pp. 462–476, 2006.

[9] M. Ding, D. Chen, K. Xing, and X. Cheng, "Localized fault-tolerant event boundary detection in sensor networks," *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE*, vol. 2, pp. 902 – 913 vol. 2, mar. 2005.

[10] J. Chen, S. Kher, and A. Somani, "Distributed fault detection of wireless sensor networks," in *DIWANS '06: Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks*, (New York, NY, USA), pp. 65–72, ACM, 2006.

[11] M. D. X. Luo and Y. Huang, "optimal fault-tolerant event detection in wireless sensor networks," in *Proc. of 13th European Signal Processing Conference*, (Antalya, Turkey), 2005.

[12] X. Luo, M. Dong, and Y. Huang, "On distributed fault-tolerant detection in wireless sensor networks," *Computers, IEEE Transactions on*, vol. 55, pp. 58 – 70, jan. 2006.

[13] R. Niu and P. K. Varshney, "Distributed detection and fusion in a large wireless sensor network of random size," *EURASIP J. Wirel. Commun. Netw.*, vol. 2005, no. 4, pp. 462–472, 2005.

[14] M. H. M. R. Niu, P. Varshney and D. Klamer, "decision fusion in a wireless sensor network with a large number of sensors,*Proc. 7th Int. Conf. Information Fusion*, ' vol. 4, pp. 861–864, June 2004.

[15] T. Clouqueur, K. Saluja, and P. Ramanathan, "Fault tolerance in collaborative sensor networks for target detection," *Computers, IEEE Transactions on*, vol. 53, pp. 320 – 333, mar. 2004.

[16] T.-Y. Wang, Y. Han, P. Varshney, and P.-N. Chen, "Distributed fault-tolerant classification in wireless sensor networks," *Selected Areas in Communications, IEEE Journal on*, vol. 23, pp. 724 – 734, apr. 2005.

[17] S. Tanachaiwiwat, P. Dave, R. Bhindwale, and A. Helmy, "Poster abstract secure locations: routing on trust and isolating compromised sensors in location-aware sensor networks," in *In SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pp. 324–325, ACM Press, 2003.

[18] J. Staddon, D. Balfanz, and G. Durfee, "Efficient tracing of failed nodes in sensor networks," in *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, (New York, NY, USA), pp. 122–130, ACM, 2002.

[19] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin, "Sympathy for the sensor network debugger," in *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, (New York, NY, USA), pp. 255–267, ACM, 2005.

[20] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentell, "Fault tolerance techniques for wireless ad hoc sensor networks," *Sensors, 2002. Proceedings of IEEE*,vol. 2, pp. 1491 – 1496 vol.2, 2002.

[21] S. Harte, A. Rahman, and K. Razeeb, "Fault tolerance in sensor networks using self-diagnosing sensor nodes," *Intelligent Environments, 2005. The IEEE International Workshop on*,pp. 7 – 12, jun. 2005.

[22] A. Sheth, C. Hartung, and R. Han, "A decentralized fault diagnosis system for wireless sensor networks,"*Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, pp. 3 pp. –194, nov. 2005.

[23] L. B. Ruiz, I. G. Siqueira, L. B. e. Oliveira, H. C. Wong, J. M. S. Nogueira, and A. A. F. Loureiro, "Fault management in event-driven wireless sensor networks," in *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and*

simulation of wireless and mobile systems, (New York, NY, USA), pp. 149–156, ACM, 2004.

[24] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: scalable coordination in sensor networks," in *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, (New York, NY, USA), pp. 263–270, ACM, 1999.

[25] L. Ruiz, T. Braga, F. Silva, H. Assuncao, J. Nogueira, and A. Loureiro, "On the design of a self-managed wireless sensor network," *Communications Magazine, IEEE*, vol. 43, pp. 95 – 102, aug. 2005.

[26] L. Paradis and Q. Han, "A survey of fault management in wireless sensor networks," *J. Netw. Syst. Manage.*, vol. 15, no. 2, pp. 171–190, 2007.

[27] M. Yu, H. Mokhtar, and M. Merabti, "Fault management in wireless sensor networks," *Wireless Communications, IEEE*, vol. 14, pp. 13 –19, dec. 2007.

[28] C.-Y. Wan, A. Campbell, and L. Krishnamurthy, "Pump-slowly, fetch-quickly (psfq): a reliable transport protocol for sensor networks," *Selected Areas in Communications, IEEE Journal on*, vol. 23, pp. 862 – 872, apr. 2005.

[29] S.-J. Park, R. Vedantham, R. Sivakumar, and I. F. Akyildiz, "A scalable approach for reliable downstream data delivery in wireless sensor networks," in *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, (New York, NY, USA), pp. 78–89, ACM, 2004.

[30] Y. Sankarasubramaniam, O. B. Akan, and I. F. Akyildiz, "Esrt: event-to-sink reliable transport in wireless sensor networks," in *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, (New York, NY, USA), pp. 177–188, ACM, 2003.

[31] Q. Han, I. Lazaridis, S. Mehrotra, and N. Venkatasubramanian, "Sensor data collection with expected reliability guarantees," *Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on*,pp. 374 – 378, mar. 2005.

[32] J. Staddon, D. Balfanz, and G. Durfee, "Efficient tracing of failed nodes in sensor networks," in *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, (New York, NY, USA), pp. 122–130, ACM, 2002.

[33] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell, "Coda: congestion detection and avoidance in sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, (New York, NY, USA), pp. 266–279, ACM, 2003.

[34] Y. Zhao, R. Govindan, and D. Estrin, "Residual energy scan for monitoring sensor networks," *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, vol. 1, pp. 356 – 362 vol.1, mar. 2002.

[35] R. Mini, A. Loureiro, and B. Nath, "The distinctive design characteristic of a wireless sensor network: the energy map," *The distinctive design characteristic of a wireless sensor network: the energy map*, Elsevier Computer Communications, vol. 27, p. 935945, 2004.

[36] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentelli *Handbook of Sensor Networks*, 2004.

[37] A. V. U. Phani Kumar, A. M. Reddy V, and D. Janakiram, "Distributed collaboration for event detection in wireless sensor networks," in *MPAC '05: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, (New York, NY, USA), pp. 1–8, ACM, 2005.

[38] J. N. AL-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: A survey," in *IEEE Wireless Communications*, vol. 11, pp. 6 – 28, dec. 2004.