Design and Analysis of Adaptive Fault Tolerant QoS Control Algorithms for Query Processing in Wireless Sensor Networks

Ngoc Anh Phan Speer

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Committee:
Dr. Ing-Ray Chen, Chair
Dr. Luiz A. DaSilva
Dr. Csaba J. Egyhazy
Dr. Chang-Tien Lu
Dr. Scott F. Midkiff

Date: April 17, 2008
Falls Church, Virginia

Keywords: Wireless sensor networks, fault tolerance, quality of service, reliability, timeliness, query processing, redundancy, energy conservation, mean time to failure

© Copyright 2008

Design and Analysis of Adaptive Fault Tolerant QoS Control Algorithms for Query Processing in Wireless Sensor Networks

Ngoc Anh Phan Speer

ABSTRACT

Wireless sensor networks (WSNs) present several unique characteristics such as resource-constrained sensors, random deployment, and data-centric communication protocols. These characteristics pose unprecedented challenges in the area of query processing in WSNs. This dissertation presents the design and validation of adaptive fault tolerant QoS control algorithms with the objective to achieve the desired quality of service (QoS) requirements and maximize the system lifetime in query-based WSNs.

Data sensing and retrieval in WSNs have a great applicability in military, environmental, medical, home and commercial applications. In query-based WSNs, a user would issue a query with QoS requirements in terms of reliability and timeliness, and expect a correct response to be returned within the deadline. Satisfying these QoS requirements requires that fault tolerance mechanisms through redundancy be used, which may cause the energy of the system to deplete quickly. We analyze the effect of redundancy on the mean time to failure (MTTF) of query-based cluster-structured WSNs, defined as the mean number of queries that a WSN is able to answer correctly until it fails due to channel faults, sensor faults, or sensor energy depletion. We show that a tradeoff exists between redundancy and MTTF. Furthermore, an optimal redundancy level exists such that the MTTF of the system is maximized.

We develop a hop-by-hop data delivery (HHDD) mechanism and an Adaptive Fault Tolerant Quality of Service Control (AFTQC) algorithm in which we utilize “source” and “path” redundancies with the goal to satisfy application QoS requirements while maximizing the lifetime of WSNs. We also compare and contrast AFTQC without acknowledgment vs. AFTQC with acknowledgement and identify conditions under which AFTQC should couple with acknowledgement to maximize the system MTTF.

To deal with network dynamics, we investigate proactive and reactive methods to dynamically collect channel and delay conditions to determine the optimal redundancy level at runtime. AFTQC can adapt to network dynamics that cause changes to the node density, residual energy, sensor failure probability, and radio range due to energy consumption, node failures, and change of node connectivity. Further, AFTQC can deal with software faults, concurrent query processing with distinct QoS requirements, and data aggregation.

We compare our design with a baseline design without redundancy based on acknowledgement for data transmission and geographical routing for relaying packets to demonstrate the feasibility. We validate analytical results with extensive simulation studies. When given QoS requirements of queries in terms of reliability and timeliness, our AFTQC design allows optimal "source" and "path" redundancies to be identified and applied dynamically in response to network dynamics such that not only query QoS requirements are satisfied, as long as adequate resources are available, but also the lifetime of the system is maximized.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1
INTRODUCTION

Over the last few years, we have seen a rapid increase in the number of applications for WSNs [1], [2]. WSNs are used in battlefield applications, and a variety of vehicle health management and condition-based maintenance applications on industrial, military, and space platforms. For military users, a primary focus has been area monitoring (security and surveillance applications). For industrial platform, WSNs are targeted for health monitoring of equipment for complex machinery and processes inside factories or on board ships. For air and space platforms, a main focus is on the overall integrated vehicle health management system for use on aircraft, rotorcraft, and spacecraft. WSNs are also used for habitat and environmental monitoring for wildlife and seabird habitat management.

1.1. Challenges in WSNs and Research Motivations

WSNs inherit most of the QoS challenges from general wireless networks. Moreover, their particular characteristics pose unique challenges as follows [3], [4].

1) Extreme resource constraints: Sensor nodes (SNs) are small-scale devices with their sizes approaching a cubic millimeter in the near future [5]. Such small devices are very limited in the energy they can store or collect from the environment. Furthermore, SNs are subject to failures due to depleted batteries or, more generally, due to environmental influences. Limited size and energy also typically means restricted resources (CPU performance, memory, wireless communication bandwidth and range). As a result, these constraints impose an essential requirement on any QoS support mechanisms in WSNs, that is, simplicity. Computation intensive algorithms, expensive signaling protocols, or overwhelming network states maintained at SNs are not feasible.

2) Unbalanced traffic: In most applications of WSNs, traffic mainly flows from a large number of SNs to a small subset of sink nodes. QoS mechanisms should be designed for an unbalanced QoS-constrained traffic.

3) High data redundancy: WSNs are characterized by high redundancy in the sensor data. However, while data redundancy helps satisfy the reliability/robustness requirement of data delivery, it may unnecessarily spend too much precious energy. Data fusion or data aggregation is a solution to maintain robustness while decreasing redundancy in the data, but this mechanism also introduces latency and complicates QoS design in WSNs.

4) Network dynamics: Node mobility, node failures, environmental obstructions, and node state transitions due to the use of power management or energy efficient scheme cause a high degree of dynamics in WSNs. This includes frequent network topology changes and network partitions. Communication failures are also a typical problem of WSNs [6]. Such a highly dynamic network greatly increases the complexity of QoS support.

5) Balancing energy consumption: In order to achieve a long-lived network, energy load must be evenly distributed among all SNs so that the energy at a single SN or a small set of SNs will not be drained out very soon. QoS support should take this factor into account.

6) Scalability: A generic wireless sensor network is envisioned as consisting of hundreds or thousands of SNs densely distributed in a terrain. Therefore, QoS support designed for WSNs should be able to scale up to a large number of SNs.

7) Multiple sink nodes: Multiple sink nodes may exist in a WSN, which impose different requirements on the network. For instance, one sink may ask SNs located in the northeast of the sensor field to send a temperature report every minute, while another sink node may only be interested in an exceptionally high temperature event in the southwest area. WSNs should be able to support different QoS levels associated with different sink nodes.

8) Heterogeneous environment: A WSN may consist of a large number of rather different SNs in terms of types, computing power, and memory. The large number raises scalability issues on the one hand, but provides a high level of redundancy on the other hand. Also, SNs have to operate unattended, since it is impossible to service a large number of SNs in remote, possibly inaccessible locations. Inclusion of heterogeneous sets of SNs raises challenges for QoS support. For instance, some applications may require a diverse mixture of SNs for monitoring temperature, pressure, and humidity, thereby introducing different reading rates at these SNs. Such a heterogeneous environment makes QoS support more challenging.

9) Multiple QoS levels: The content of data or high-level description reflects the criticality of the real physical phenomena and is thereby of different priority with respect to the

quality of the applications. QoS mechanisms may be required to differentiate data importance and set up a priority structure.

This dissertation research is motivated by the challenges in the WSNs. We develop a Hop-by-Hop Data Delivery (HHDD) protocol that utilizes path and source redundancies to achieve the required QoS requirement in terms of reliability and timelines while maximizing the system lifetime. Our Adaptive Fault Tolerant QoS Control (AFTQC) algorithm, built on top of the HHDD protocol, addresses the issues of resource constraints, unbalanced traffic, high data redundancy, network dynamics, balancing energy consumption, scalability, multiple sink nodes, and multiple QoS levels. We consider a homogeneous WSN with SNs being deployed massively. The AFTQC algorithm supports multiple QoS levels by using different levels of redundancy to answer queries with different QoS requirements. It does not require an initial route setup for data delivery in order to save energy of WSNs. This algorithm is simple enough to be executed on resource constrained WSNs. It supports concurrent queries from multiple sink nodes. It is also scalable to a large WSN and adaptive to network dynamics. The optimal level of redundancy used to answer a query is dynamically adjusted when the sensor density is changed due to channel failures, sensor failures or sensor energy depletion.

1.2. Wireless Sensor Network Applications

SNs can be used in many applications for continuous sensing, event detection, location sensing, and local control of actuators. The applications in WSNs can be categorized into military, environmental, medical, home, and commercial applications.

1) Military applications: WSNs can be an integral part of military command and control, communications, computing, intelligence, surveillance, reconnaissance and targeting systems. The rapid deployment, self-organization and fault tolerance characteristics of WSNs make them a very promising sensing technique for military. Some of the military applications of WSNs are monitoring forces, equipment and ammunition; battlefield surveillance; reconnaissance of opposing forces and terrain; targeting; battle damage assessment; and nuclear, biological and chemical attack detection and reconnaissance [1], [2]. For example, A Patrol, Search and Rescue vehicle (PSAR) uses information provided by a WSN to patrol the border, detecting illegal border crossings, drug trafficking, and other criminal activities. Also, a rescue mission in this

inhospitable terrain may involve PSAR(s) navigating the terrain, interacting continuously with the WSN, tapping into the stored information to chart safe paths. The WSN informs the PSAR in real-time of sources of imminent danger, the presence of trapped or wounded people, etc. The nature of these interactions requires the WSN to provide timely, high-quality information.

Our algorithm can be applied to some of these applications which are query-based WSN applications. For example, for monitoring forces, equipment and ammunition in a battlefield, every troop, vehicle, equipment and critical ammunition can be attached with small SNs that report the status. These reports are gathered at the sink nodes and sent to the troop leaders. Critical terrains, approach routes, paths and strait can be rapidly covered with WSNs for battlefield surveillance and reconnaissance. WSNs can be deployed in the target areas to gather the battle damage assessment data. WSNs can also be used for detailed reconnaissance after a nuclear, biological and chemical attack is detected without exposing human to nuclear radiation.

2) Environmental applications: Some environmental applications of WSNs include pollution monitoring; wildlife and habitat management; monitoring environmental conditions; chemical and biological detection; precision agriculture; biological, earth, and environmental monitoring in marine, soil, and atmospheric contexts; meteorological or geophysical research; forest fire detection; flood detection; and bio-complexity mapping of the environment [7] - [22].

One of the query-based WSN application examples that our algorithm can be applied to is a pollution monitoring and surveillance system used in many power generation plants that use coal as a major energy source [7]. Such applications can collect real-time air and water quality data to satisfy state government's air-water pollution monitoring requirements. An example of flood detection is the ALERT system [20] deployed in the US, which consists of several types of SNs such as rainfall, water level and weather sensors. For bio-complexity mapping of the environment, an example is the Terrestrial Ecology Observing Systems implemented by the James Reserve in Southern California and UCLA Center for Embedded Networked Sensing [21], [22].

3) Medical applications: Some of the medical applications for sensor networks are providing interfaces for the disabled; integrated patient monitoring; diagnostics; drug administration in hospitals; monitoring the movements and internal processes of insects or other small animals; telemonitoring of human physiological data; and tracking and monitoring doctors and patients inside a hospital [23] - [29].

WSNs can monitor and detect elderly people's behavior. This allows doctors to identify pre-defined symptoms earlier, and facilitate a higher quality of life for the patients. A ''Health Smart Home'' is designed in the Faculty of Medicine in Grenoble - France [23] to validate the feasibility of such system. The Health Smart Home system uses a WSN for monitoring of patients at home or in a long-term healthcare center. Smart wireless sensors are placed in different rooms to tract patient movements to detect a fall or inactivity of elderly people. Every second, a number of queries are invoked to gather movement information. One query is to gather information to display a synoptic of instantaneous positions and situations of the patients. Another query is to get a chronological display of successive events. The last one is to get the time occupation for each room. This information received can be processed locally so the on-call nurse or doctor may directly know about their patient statuses. This data can also be transmitted to a remote medical data center for monitoring. This type of applications requires the WSN to support concurrent queries and provide real-time and high quality information. Patients can have SNs that identify their allergies and required medications to minimize the chance of getting and being prescribed the wrong medication. Computerized systems as described in [28] have shown that they can help minimize adverse drug events. Examples of query-based WSN applications that our algorithm can be applied to are WSNs that collect human physiological data for medical exploration.

4) Home applications: Some of the home applications are home automation and smart environment [30] – [32]. For home automation, SNs and actuators can be buried in appliances, such as vacuum cleaners, ovens, refrigerators, etc. Their ability to interact with each other and with the external network via the Internet or Satellite allows end users to manage home devices locally and remotely more easily. Some examples of smart environments are the ''Residential Laboratory'' at Georgia Institute of Technology and the smart environment described in [31]. SNs can be embedded into furniture and appliances, and communicate with each other and the room server, which control the services offered such as printing, scanning, and faxing.

5) Commercial applications: Some of the commercial applications are monitoring material fatigue; managing inventory; monitoring product quality; constructing smart office spaces; environmental control in office buildings; robot control and guidance in automatic manufacturing environments; interactive toys; interactive museums; factory process control and automation; monitoring disaster area; smart structures with SNs embedded inside; machine

diagnosis; transportation; factory instrumentation; local control of actuators; detecting and monitoring car thefts; vehicle tracking and detection; and instrumentation of semiconductor processing chambers, rotating machinery, wind tunnels, and anechoic chambers [33] – [36]

An example of interactive museums [33] is the San Francisco Exploratorium that features a combination of data measurements and real time cause-and-effect experiments which facilitates children mental development through touch and speech. SNs can also be deployed to detect and identify car thefts within a geographic region and report these to remote end users by the Internet [35].

Example query-based WSN applications that our algorithm can apply to are environmental control in office buildings and inventory control. A distributed WSN system can be installed to control the air flow and temperature in different parts of the building which can substantially reduce the building energy consumption. For inventory control, each item in a warehouse may have a SN attached to allow the end user to track and locate them as well as count the number of items in the same category. Another example is the accident monitoring system. In the case of emergency, wireless sensor networks can be used to collect the status information of people in the accident and the surrounding information in the accident spot. The systems should sense and analyze the information and inform users as correct and detailed as possible within the limited time [36].

Most query-based applications in WSNs are interactive, query-specific delay sensitive, mission critical and non-end-to-end applications. These applications need to receive the desired data as quickly as possible and as reliably as possible. Our algorithm aims to satisfy the requirements of query-based applications. With the use of path and source redundancy, the algorithm satisfies the user specific QoS requirements for each query in terms of reliability and timeliness while maximizing the system lifetime in term of the number of queries the system can service. Therefore, our approach is applicable to a wide range of WSN applications.

1.3. Research Statement

Fault tolerance to deal with sensor faults (including sensor measurement faults) in WSNs is an emerging field. This dissertation aims to design and analyze adaptive fault tolerant QoS control algorithms by incorporating redundancy to allow sensor and communication faults to be tolerated, with the goal of maintaining the overall functioning of the system in the presence of

faults, while prolonging the system lifetime of WSNs. We propose to address this research issue by considering two different types of redundancy, namely, "source" and "path". For an application with specified QoS requirements in terms of reliability and timeliness (response time), we show that these mechanisms can be used to achieve reliability and timeliness effectively and efficiently.

Our algorithms can deal with the case in which multiple queries, each with distinct fault tolerance and timeliness QoS requirements, are being processed concurrently. The best level of redundancy depends on the density of nodes, energy of nodes, transmission/node failure probabilities, transmission speed, and node connectivity, which will be dynamically changed as the environment changes due to network dynamics such as SN failures. We aim to design adaptive fault tolerance QoS control algorithms that can determine the best redundancy level employed at runtime to satisfy the QoS requirements while maximizing the lifetime of WSNs in the presence of network dynamics.

1.4. Related Work

In this section, we survey related work in the areas of fault tolerance, quality of service, energy conservation and routing in WSNs.

1.4.1. Fault Tolerant WSNs

Applications such as security and surveillance monitoring, battlefield command and control, and wildlife or medical monitoring rely on the correct functioning of the underlying WSNs for data sensing and retrieval in response to application queries. In such applications, the WSNs are often deployed in an area where replacements of sensors are difficult or impossible. We consider three major sources of faults that could cause a WSN to fail. One source is due to *energy depletion* of SNs, such that the underlying WSN simply exhausts its energy to be able to answer queries. Another source is due to *sensor faults* including measurement faults. The third source is due to *communication faults* because of noise and interference in the WSN.

To conserve energy of SNs, a well accepted approach is for the WSN to self-organize itself into clusters. Within a cluster, a cluster head is elected to perform more data aggregation and relay duties than normal SNs and is rotated among SNs in the cluster for balancing energy consumption. An intra-cluster routing tree is maintained dynamically in response to sensor faults.

For long-haul networks, cluster heads may relay data cluster-by-cluster to reach a processing center (e.g., where a user query is issued). Moreover, for applications concerned with sensor readings such as the minimum/maximum/average of sensor data, cluster heads can also perform in-network data aggregation and compression functions to reduce energy consumption [40]. A cluster can be reconfigured globally by the system for optimization purposes or in a distributed manner on a per-sensor-node basis. In [37], a cluster is defined around each SN covering a sensor node's maximum transmission radius area, and a shortest path routing table is maintained by each node to reach its neighbors in the cluster. After receiving a query, a SN transmits responses to the requesting node in its cluster through the shortest path in multiple "minimum transmission energy" hops. If the shortest path fails due to sensor faults, it can directly transmit responses to the requesting node in a single "maximum transmission energy" hop, at the expense of energy consumption.

To cope with the second source of faults, i.e., sensor faults, a general approach is to incorporate redundancy to allow sensor faults to be detected, isolated, and corrected so that the system can continue to function correctly in data sensing and retrieval. However, the use of redundancy impacts the energy consumption rate of the system since more SNs would need to be used as redundancy to achieve sensor fault tolerance. Therefore, there is a tradeoff between these two sources of faults. On the one hand, we like to incorporate redundancy to deal with sensor faults. On the other hand, redundancy should be used only as needed so as not to quickly deplete the energy of the system.

Current research work on fault tolerance mechanisms to cope with sensor faults in WSNs can be classified into hardware redundancy, time redundancy and information redundancy [38] [42]. Hardware redundancy utilizes extra hardware for fault detection or masking. For example, two temperature sensors can be used to agree on a temperature reading before the reading is considered as a correct response. If a discrepancy exists, then a third temperature sensor can be used to arbitrate through voting. A sensor can also be made to disambiguate a sensor measurement fault from a true event by using a distributed Bayesian algorithm [40], [41] after comparing readings obtained from its neighbor sensors of the same type. At the processing center (i.e., user query) end, more than one sensor reading responding to a query can be propagated back to the processing center via multiple paths, from which a voting can be performed using the first three readings. A majority reading, if it exists, will be passed to the

application as the legitimate response. This approach was suggested in [39] for data fault tolerance of data propagation in WSNs. The time to live (TTL) value of query and reply packets is adjusted to allow multiple readings to return to the processing center through multiple paths.

Time redundancy is a simple form of fault tolerance that utilizes repeated execution as the primary mechanism. One can monitor the output of a sensor reading query to see if the output returned is within a normal range. If the reading is out of ordinary (e.g., an unusually high reading), a second reading query can be performed with the output possibly returned through a different path. A comparison then is made to detect if the reading reflects the actual environment reading.

Information redundancy uses the relationship among sensor data from the physical world for fault detection. For example, a relation exists between speed, pressure and position in a diesel engine such that if the pressure sensor is detected to be faulty, one can deduce its value from the other two sensor readings. By means of a lookup table that relates one variable with others, an extrapolation method has been used in [38] to tolerate a sensor measurement fault. Information redundancy in the form of adopting different classification algorithms was employed in the presence of sensor faults [42], thus allowing fewer backup sensors to be used for fault tolerance. Suppose that there are originally 4 height and 2 color sensors used to classify objects. Then, a second classification algorithm can be used if there are only 3 height and 2 color sensors available, and a third classification algorithm can be used if there are only 4 height and 1 color sensors available. Thus, compared with hardware redundancy that would require the use of extra hardware to tolerate sensor faults, information redundancy is more economical.

We consider hardware redundancy in this dissertation, specifically, path redundancy [39] - [44] and source redundancy to cope with sensor faults. Time redundancy is not considered because queries normally carry with them a deadline, which makes time redundancy not suitable for our target applications. Information redundancy is not considered because it requires specific knowledge of the physical law governing sensor measurements, thus making it difficult to be used as generic fault tolerance mechanisms to satisfy QoS requirements of applications. We consider a WSN as having experienced a failure when it fails to deliver sensor data correctly in response to an application-level query, due to one of the three sources of faults, i.e., energy depletion, sensor fault, or communication fault.

1.4.2. Quality of Service

From the network QoS perspective, we consider how the underlying communication network can deliver sensor data while efficiently utilizing network resources. A WSN can be source-driven or query-based depending on the data flow. In source-driven WSNs, sensors initiate data transmission for observed events to interested users, including possibly reporting sensor readings periodically. An important research issue in source-driven WSNs is to satisfy QoS requirements of event-to-sink data transport while conserving energy of WSNs. ESRT [45] has been proposed to address this issue with reliability as the QoS metric. The goal of ESRT is to achieve reliable event detection in WSNs with minimum energy expenditure. It includes a congestion control component that serves the dual purpose of achieving reliability and conserving energy. The algorithm of ESRT mainly runs on the sink with minimal functionality required at resource constrained SNs. A sink would estimate the event-to-sink reliability and adjusts the reporting frequency of sensor nodes to achieve the desired reliability. This dissertation research concentrates on query-based rather than source-driven WSNs. We consider both reliability and timeliness as the QoS requirement. Further, while ESRT aims to provide end-to-end reliable transport between a sink-source pair by means of congestion control exercised by the sink node based on feedback, our hop-by-hop data delivery protocol is designed to propagate data hop by hop to satisfy the QoS requirements without feedback in order to conserve energy and better satisfy a query deadline requirement.

In query-based WSNs, queries and data are forwarded to interested entities only. A user would issue a query with QoS requirements in terms of reliability and timeliness. Retrieving sensor data such that QoS requirements are satisfied is a challenging problem and has not been studied until recently [46] – [56]. The general approach is to apply redundancy to satisfy the QoS requirement. In this dissertation we are also interested in applying redundancy to satisfy application specified reliability and timeliness requirements for query-based WSNs. Moreover, we determine the optimal redundancy level that could satisfy QoS requirements while prolonging the lifetime of the WSNs. Specifically; we develop the notions of “path” and “source” level redundancies. When given QoS requirements of a query, we identify optimal path and source redundancies such that not only QoS requirements are satisfied, but also the lifetime of the system is prolonged. In particular, in order to eliminate energy expended for maintaining routing

paths in WSNs, we develop a hop-by-hop data delivery protocol to achieve the desired level of redundancy.

Existing end-to-end QoS solutions for WSNs are based on the concept of end-to-end QoS requirements such as per-packet delay, loss probability, reliability, etc. The problem is that it may not be feasible to implement end-to-end QoS in WSNs due to the complexity and high cost of the protocols for resource constrained sensors. An example is Sequential Assignment Routing (SAR) [47] that utilizes path redundancy in dense WSNs from a source node to the sink node. Each sensor uses a SAR algorithm for path selection. It takes into account the energy and QoS factors on each path, and the priority level of a packet. For each packet routed through the network, a weighted QoS metric is computed as the product of the additive QoS metric and a weight coefficient associated with the priority level of that packet. The objective of the SAR algorithm is to minimize the average weighted QoS metric throughout the lifetime of the network. The algorithm does not consider the reliability issue. Our work considers the use of multiple paths instead of selecting one path to satisfy QoS requirements including reliability and timeliness.

A QoS routing protocol (SPEED) [48] has been proposed to provide soft real-time end-to-end timeliness guarantee. The protocol requires each node to maintain information about its neighbors and exploits geographic forwarding to find the paths. In addition, SPEED strives to ensure a certain speed for each packet delivery so that each application can estimate the end-to-end delay for the packets by considering the distance to the sink and the speed of the packet delivery before making the admission decision. Using the distance and delay, each node evaluates the packet progress speed of each neighbor node and forwards a packet to a node whose progressive speed is higher than the pre-specified lower-bound speed. All mechanisms work in a localized way, hence, SPEED is scalable. However, the SPEED protocol does not provide any guarantee in packet reliability. In our approach we adopt the speed concept to satisfy a query's timeliness requirement. In addition, we consider packet reliability as one of the QoS requirements. We also adopt hop by hop routing based on geographic routing as in SPEED to reduce the latency and energy consumption. However, we use broadcasting to implement multi-path routing to satisfy the reliability requirement, instead of unicasting in SPEED for single-path routing.

ReInForM [49] has been proposed to address end-to-end reliability issues. ReInForm considers information awareness and adaptability to channel errors, along with a differentiated allocation strategy of network resources based on the criticality of data. The protocol sends multiple copies of a packet along multiple paths from the source to the sink such that data is delivered with the desired reliability. It uses the concept of dynamic packet state in the context of sensor networks to control the number of paths required for the desired reliability, based on local knowledge of the channel error rate and topology. The protocol observes that for uniform unit disk graphs, the number of edge-disjoint paths between nodes is equal to the average node degree with a very high probability. This protocol results in the use of the disjoint paths in a thin band between the source and the sink. However, the protocol only concerns QoS in term of reliability and does not consider the energy issue. Our algorithm considers both reliability and timeliness as QoS requirements. Also, rather than just sending multiple copies of a packet through multiple paths to achieve the required reliability, our algorithm determines the optimal number of redundant paths and sources to achieve the required QoS requirements and also to maximize the lifetime of WSNs.

In [50], M. Perillo et al. propose a method for maximizing the lifetime of WSNs while satisfying a minimum level of reliability. This maximization is achieved through the joint optimization of scheduling active sensor sets and finding paths for data routing. The lifetime is defined as the sum of the time that all sensor sets are used. The approach uses the strategy of turning off redundant sensors for a period of time to save energy while considering the tradeoff between energy consumption and reliability. This approach can extend the lifetime of a network considerably compared with approaches that do not use intelligent scheduling. However, this approach is applicable to source-driven one-hop WSNs and is limited to application reliability only. Our work considers both reliability and timeliness requirements and is designed for query-based multi-hop WSNs for which it is impractical to determine the sensor sleep time because queries may arrive at the system at any arbitrary time.

Recently, a multi-path and multi-speed routing protocol called MMSPEED [51] is proposed which takes into account both timeliness and reliability as QoS requirements. The goal is to provide QoS support that allows packets to choose the most proper combination of service options depending on their timeliness and reliability requirements. For timeliness, multiple QoS levels are supported by providing multiple data delivery speed options. For reliability, multiple

reliability requirements are supported by probabilistic multi-path forwarding. The protocol provides end-to-end QoS provisioning by employing localized geographic forwarding using immediate neighbor information without end-to-end path discovery and maintenance. MMSPEED adapts to network dynamics such as channel error conditions and speed changes to determine the number of forwarding nodes (thus forming multiple paths) in each hop to satisfy the overall reliability and timeliness QoS requirements. However, MMEPEED does not consider the energy issue; hence, it is only applicable for short-lived WSN applications with a mission duration lasting only a few hours or at most one day. Our work adopts hop-by-hop geographical routing and deals with such environment changes in a similar way. Our hop-by-hop data delivery protocol considers energy consumption, in addition to reliability and timeliness requirements as in MMSPEED. Moreover, we determine the optimal path and source redundancy to achieve the required QoS and maximize the system lifetime. In contrast to MMSPEED in which each SN is responsible for determining the number of forwarding nodes, in our approach the determination of the optimal redundancy level is performed by the sink node, thereby reducing the energy consumption of SNs. Further, we also consider network dynamics due to sensor failures, energy depletion and sensor connectivity.

In [52], QoS is defined as the optimum number of sensors that should be sending information to the sinks at any given time. The protocol utilizes the base station to communicate QoS information to each of the sensors using a broadcast channel. It exploits the mathematical paradigm of the Gur Game to dynamically adjust to the optimum number of sensors. The objective is to maximize the lifetime of WSNs by having sensors periodically power-down to conserve energy, and at the same time having enough sensors powered-up and sending packets to the sinks to collect enough data. The protocol allows the base station to dynamically adjust the QoS resolution. This solution requires the determination of the amount of sensors that should be powered up a priori to maintain a resolution. QoS metrics for data delivery such as reliability and timelines are not considered.

In [53], B. Deb et al. discuss the problem of efficient information assurance in WSNs where the assurance level of information is defined as the probability of information delivery (desired reliability) to the sink. The proposed schemes utilizes hop-by-hop broadcast for information delivery to reduce packet overhead while still attaining the desired reliability. The first proposed protocol is the Hop-by-Hop Broadcast Protocol which does not use

acknowledgments. At each hop, nodes that decide to forward will broadcast *n* copies of the packet. The second one is the Hop-by-Hop Broadcast with Acknowledgment which uses acknowledgement and a stop-and-wait protocol to forward packets. The number of copies of a packet to be sent at each hop is determined based on the desired reliability and the channel error. However, these protocols only consider the reliability issue and do not consider source redundancy or system lifetime. Our algorithm considers reliability and timeliness as the QoS requirements and takes into account channel error, sensor fault, as well as sensor energy depletion in determining the optimal number of paths and sources to achieve the desired QoS requirements and maximize the system lifetime. Our hop-by-hop data forwarding protocol derives from the hop-by-hop broadcast protocol without acknowledgement to save packet delivery cost and conserve energy, except that the decision to form *m* paths is determined by a sink node when answering a query so as to satisfy QoS requirements while maximizing the system lifetime. Also, the protocol allows *m* redundant paths to be implemented while data is being forwarded from a specified source to a specified destination node.

1.4.3. Energy Conservation in WSNs

One of the characteristics of WSNs is severe resource constraints. The constraints on resources involve energy, bandwidth, memory, buffer size, processing capability, and limited transmission power. Among them, energy is a primary concern since energy is severely constrained at SNs and it may not be feasible to replace or recharge the battery for SNs that are often expected to work in a remote or inhospitable environment. Several techniques and protocols have been proposed to address this issue.

Data fusion or data aggregation is a solution to reduce energy consumption by decreasing redundancy in the data. Directed diffusion [44] relies on local interactions among nodes to create efficient paths for data flow. No global routing state is kept anywhere in the system. Each node chooses its own sources from which to receive data, leading to reasonably efficient data propagation at a global level. The protocol achieves energy savings by allowing intermediate nodes to aggregate responses to queries. To indicate the overall lifetime of SNs, the protocol uses a metric called Average Dissipated Energy to measure the ratio of total dissipated energy per node to the number of event seen by sinks.

Using multiple transmission power levels at SNs is another technique to reduce energy consumption. In [37], the authors present a protocol called Shortest Path Minded SPIN (SPMS) that efficiently disseminates information among sensors in an energy-constrained WSNs. Nodes name their data using high-level data descriptors, called meta-data. They use meta-data negotiations to eliminate the transmission of redundant data throughout the network. The protocol achieves additional energy saving by allowing sensors nodes to operate at multiple power levels.

PEGASIS (Power-Efficient Gathering in Sensor Information Systems) [57] forms a chain passing through all nodes where each node receives from and transmits to the closest possible neighbor to reduce the number of nodes communicating directly with the base station. Data is collected starting from each endpoint of the chain until the randomized head node is reached. Data is fused each time it moves from node to node and the final data is transmitted to the based station by the head node.

PEDAP (Power Efficient Data gathering and Aggregation Protocol) [58] uses a minimum spanning tree rooted at the base station to prolong the lifetime of the system. A data gathering round is defined as the process of gathering all the data from SNs to the base station. In these protocols, the lifetime of the system is calculated by the total number of data gathering rounds utilizing the following metrics - First Node Die (FND), Half Node Die (HND) and Last Node Die (LND).

Clustering [59] – [63] is a widely accepted technique for reducing energy consumption in WSNs. In order to achieve a long-lived network, energy load must be evenly distributed among all SNs so that the energy at a single SN or a small set of SNs will not be depleted too rapidly. Clustering prolongs the system lifetime of WSNs because it reduces contention on wireless channels [77] and supports data aggregation and forwarding at cluster heads (CHs). HEED (Hybrid Energy-Efficient Distributed) [59] increases energy efficiency by periodically rotating the role of CH among SNs with equal probability such that the SN with the highest residual energy and node proximity to its neighbors within a cluster area is selected as a CH. In LEACH (Low-Energy Adaptive Clustering Hierarchy) [60], the key idea is to reduce the number of nodes communicating directly with the base station by forming a small number of clusters in a self-organizing manner. LEACH uses randomization with equal probability in cluster head selection to achieve energy balance. REED (Robust Energy Efficient Distributed) [61] considers the use of

redundancy to cope with failures of CHs in hostile environments. Fault tolerance can be achieved by selecting k independent sets of CHs on top of the physical network, so that each SN can quickly switch to other cluster heads in case of failures or attacks on its current CH.

Hierarchical clustering techniques can aid in reducing energy consumption in WSNs. An energy efficient hierarchical clustering approach is presented in [64] in which SNs are organized into a hierarchy of clusters using a distributed randomized algorithm. The algorithm minimizes the total energy spent in the system to communicate information gathered by all sensors to a processing center. The results show that energy savings increase with the number of levels in the hierarchy.

In this dissertation, we consider clustered-based WSNs. We adopt a clustering algorithm similar to LEACH or HEED to rotate the role of CH among SNs to balance energy consumption. The energy consumption due to periodic clustering is taken into consideration when we compute the MTTF of the WSN in our analysis. Our approach of satisfying application reliability and timeliness requirements while maximizing the system lifetime is to determine the optimal level of redundancy at the "source" and "path" levels. The source level redundancy refers to the use of multiple sensors within a cluster to return the requested sensor reading to return to the CH. The path level redundancy refers to the use of multiple paths to relay the reading from a source CH to the sink node. Since senor networks are constrained with resources, instead of incurring extra overhead to formulate multiple paths before data delivery, we develop a hop-by-hop data delivery protocol to dynamically form multiple paths during data delivery.

1.4.4. Routing in WSNs

Routing in WSNs can be divided into *flat-based* routing, *hierarchical-based* routing, and *location-based* routing depending on the network structure [65]. In flat-based routing, all SNs are typically assigned equal roles or functionality. In hierarchical-based routing, SNs will play different roles in the network. In location-based or geographic-based routing, SNs positions are exploited to route data in the network.

In this dissertation research, we adopt geographical routing based on location awareness to reduce data packet forwarding cost and energy consumption. Many recent geographical routing protocols have been proposed that are applicable to WSNs [66] - [71]. SNs 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 [72], [73]. Alternatively, the location of SNs may be available directly by communicating with a satellite using GPS if nodes are equipped with a small low-power GPS receiver [66], [67] or distributed location services [70], [74]. In our model, we also exploit localized packet routing without end-to-end path setup and maintenance. The localized geographic routing has the following three advantages in WSNs: (1) scalability to a very large and dense sensor network; (2) no path setup and recovery latency, hence suitable for both critical aperiodic and periodic packets; and (3) per-packet path discovery resulting in self adaptation to network dynamics.

Geographic Adaptive Fidelity (GAF) [67] is an energy-aware location-based routing algorithm that is applicable to WSNs. The network area is first divided into fixed zones and forms a virtual grid. Inside each zone, SNs collaborate with each other to play different roles. For example, one SN will be elected to stay awake for a certain period of time, and then the rest go to sleep. This SN is responsible for monitoring and reporting data to the base station on behalf of the SNs in the zone. Hence, GAF conserves energy by turning off unnecessary SNs in the network without affecting the level of routing fidelity. Each SN uses its GPS-indicated location to associate itself with a point in the virtual grid.

Yu et al. [68] discuss the use of geographic information while disseminating queries to appropriate regions since queries often include geographic attributes. The protocol, Geographic and Energy Aware Routing (GEAR), uses neighbor selection heuristics based on energy and location awareness to route a packet toward the destination region. To conserve energy, it restricts the number of interests in directed diffusion by only considering a certain region rather than sending the interests to the whole network.

Stojmenovic et al. [71] described localized multipath geographical routing schemes which are a variant of Geographic Distance Routing (GEDIR) [69]. In these protocols, a source node will forward a message to c best neighbors according to a certain criterion, and at intermediate nodes, it is forwarded to only the best neighbor. In the disjoint c-GEDIR method, each intermediate node, upon receiving the message, will forward the message to its best neighbor among those who never received the message. Thus, in effect, the methods attempts to create c disjoint paths. This protocol is applicable to WSNs; however, it does not discuss QoS or energy issues. In our model, we also form multiple paths. However, we use hop-by-hop

broadcasting rather than multicasting to achieve multipath routing to reduce the overhead of keeping neighbor status.

GPSR (Greedy Perimeter Stateless Routing) [66] uses the positions of routers and a packet's destination to make packet forwarding decisions. GPSR makes greedy forwarding decisions using only information about a router's immediate neighbors in the network topology. When a packet reaches a region where greedy forwarding is impossible, the algorithm recovers by routing around the perimeter of the region. By keeping state only about the local topology, GPSR scales better in per-router state than shortest-path and ad-hoc routing protocols as the number of network destinations increases. Under mobility's frequent topology changes, GPSR can use local topology information to find new routes quickly. In this dissertation research, we utilize GPSR for geographical routing. Our hop-by-hop data delivery protocol is built on geographical routing and uses hop-by-hop routing based on location information to make routing decision without end-to-end path setup and maintenance.

1.5. Thesis Contribution

In a query-based WSN, the system must perform data sensing and retrieval and possibly aggregate data as a response at runtime. Since a WSN is often deployed unattended in areas where replacements of failed sensors are difficult, energy conservation is of primary concern. While the use of redundancy is desirable in terms of satisfying QoS to cope with sensor faults, it may adversely shorten the lifetime of the WSN, as more SNs will have to be used to answer queries, causing the energy of the system to drain quickly.

We analyze the intrinsic tradeoff between fault tolerance and energy conservation for satisfying the QoS requirements of queries while prolonging the lifetime of WSNs designed to answer user queries. We define the system failure as the inability of the system to answer queries due to either sensor/channel faults or energy depletion. By means of a probability model, we show that while using path and source redundancies could increase the probability that data are delivered reliably, there is a tradeoff in reliable data delivery vs. energy consumption. We demonstrate that there exists an optimal level of redundancy that should be used by the system in order to maximize the mean time to failure, when given a set of parameter values characterizing the WSN and workload environment. Once the optimal path and source redundancy levels are

determined by the system designer at design time, they can be deployed in the WSN to prolong the lifetime of the system.

We develop a path and source redundancy fault tolerance mechanism, which, when properly employed, could achieve QoS requirements while maximizing the lifetime of query-based WSNs. We discuss how this mechanism can be realized using hop-by-hop packet broadcasting and propose a hop-by-hop data delivery protocol to implement it. We analytically derive the probability of successful data delivery within a real-time constraint, as well as the amount of energy consumed per query. We design an adaptive fault tolerance QoS control algorithm for determining the best redundancy level employed at runtime to satisfy the QoS requirements of queries while maximizing the lifetime of WSNs. This algorithm is adaptive to network dynamics, including changes to the density of nodes, energy of nodes, transmission/node failure probabilities, transmission speed, and node connectivity induced by node failures and environment changes. We extend the model to handle the case where multiple queries, each with distinct fault tolerance and timeliness QoS requirements, are being processed concurrently. We also extend the model to handle the possibility of software faults and data aggregation in a WSN.

To deal with network dynamics, we investigate proactive and reactive methods for the system to dynamically collect channel and delay conditions to determine optimal redundancy at runtime. We also design mechanisms to adapt to status changes of sensor nodes due to energy consumption and node failures. We analyze our proposed adaptive fault tolerant QoS control algorithm and validate it with simulation studies based on J-Sim. We compare our algorithm with a baseline WSN system based on acknowledgement and geographical routing and demonstrate the feasibility and benefits of our algorithm in prolonging the system lifetime while satisfying the QoS requirements of queries.

1.6. Thesis Organization

The rest of the research defense dissertation is organized as follows. In Chapter 2 we present the system model and system assumptions used in the dissertation. We also define the energy model used in our analysis and define the system Mean Time to Failure (MTTF) metric. In Chapter 3, we describe our proposed hop-by-hop data delivery (HHDD) protocol and its specific implementation using two different approaches – proactive and reactive. We analyze the

pros and cons of each approach. We develop an adaptive fault tolerant QoS control algorithm (AFTQC) that can adapt to environmental changes and support concurrent queries. This algorithm drives the execution of the HHDD protocol. In Chapter 4, we develop probability models for the assessment of QoS and lifetime properties of WSNs operating under our AFTQC algorithm. We also discuss the applicability of AFTQC for applications with real-time deadlines running in query-based WSNs. In Chapter 5, we present numerical data demonstrating the existence of optimal redundancy for maximizing the MTTF of the system while satisfying QoS requirements, with proper physical explanations given. We also illustrate how AFTQC reacts with network dynamics. In Chapter 6, we discuss the simulation environment and perform extensive simulation results to validate the analytical results obtained in Chapter 5. Chapter 7 summarizes the dissertation research and outlines some future research areas. Appendix A lists the acronyms and notations for symbols used in the dissertation.

Chapter 2
SYSTEM MODEL

This chapter presents the system model for a query-based WSN. We discuss the assumptions made in the system model. We also define the system mean time to failure and quality of service (QoS) requirements used in our model.

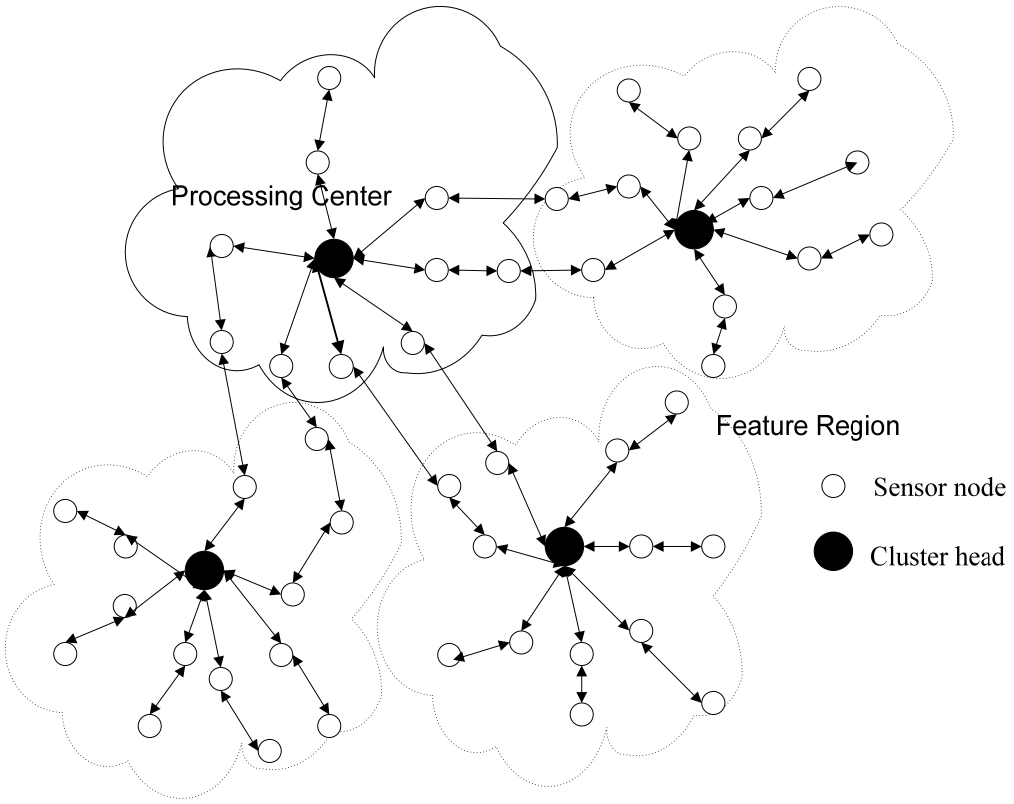

Figure 2-1: Cluster-Based WSNs.

A WSN consists of a set of low-power SNs typically deployed through an air-drop into a geographical area. We make the following assumptions regarding the structure and operation of a query-based WSN:

1. SNs are indistinguishable with the same initial energy level E_o.

2. SNs are deployed into a geographical rectangular area of size AB with sides of length A and B. This assumption has been used in the literature [59], [60], [64] to simplify the analysis although the method developed in this paper can deal with other geographical shapes.
3. SNs are distributed according to a homogeneous spatial Poisson process with intensity λ which has the physical meaning of the expected number of sensors in a square unit. We use the assumption of Poisson process for analytical convenience because it allows us to derive the average distance between a sensor and its cluster head [76]. In the simulation we first validate the analytical model with the Poisson distribution. Then we compare simulation results under both uniform and Poisson distributions and verify that the results are insensitive to the distribution used. The sensor density changes dynamically as a function of time. We use the symbol λ to generally refer to the node density at a particular point in time and $\lambda(t)$ to refer to the node density at time t.
4. The failure behavior of a SN due to environment conditions (i.e., harsh environments causing hardware failure) is characterized by a failure probability parameter q (where $0<q<1$). This parameter is assumed to be a constant.
5. A clustering algorithm like HEED [60] or LEACH [64] is employed to organize sensors into clusters for energy conservation purposes, as illustrated in Figure 2-1. A CH is elected in each cluster. The functions of a CH are to manage the network within the cluster, gather sensor reading data from the SNs within the cluster, and relay data in response to a query. The algorithm is incremental in nature and converged after a number of iterations when all sensors find their clusters. The clustering algorithm is executed periodically by all SNs in iterations in which:
    - A SN announces its role as a CH candidate with probability p.
    - The announcement message carrying the candidate CH's residual energy information is broadcast with the time to live (TTL) field set to the number of hops bounded by the cluster area size predetermined at static time.
    - Any non-CH SN overhearing the announcement can select a CH with the highest residual energy to join a cluster.

- This announcement and join process is executed in iterations such that a tentative CH can change its role to a SN if it overhears a CH candidate having a higher residual energy in a subsequent iteration.
- If a tentative CH does not hear any CH announcement, p is doubled in the next iteration.

A clustering algorithm as described above can be proven to converge within a finite number of iterations and in effect could randomly rotate the role of a CH among SNs in a cluster with probability p so that sensors consume their energy evenly [60]. This probability parameter p depends on the cluster area size, i.e., if a feature cluster area size is A_c then the average number of sensors in this cluster would be λA_c. In general let n_s denote the number of sensors in a cluster. Then p would be set to $1/n_s$ to effect this fair CH rotation among sensors in the cluster. Note that in order to deal with uneven SN distribution, the probability parameter p doubles in the next iteration until it becomes 1, so in the worst case when a SN cannot find any CH to join a cluster, it will eventually form a cluster by itself with probability 1. This unbalanced clustering behavior occurs rarely when the WSN is dense. We assume that the WSN deployed is sufficiently dense to satisfy the connectivity condition proven in [75] so that sensors within a cluster are well connected. When the WSN is sufficiently dense and the target cluster area size is the same, it is shown that clusters are balanced in practice [60]. The total energy expended by the system depends on the period ($T_{clustering}$) over which the clustering algorithm is executed and the energy expended per execution ($E_{clustering}$). The clustering algorithm is assumed to be executed as often as possible (with the rate of $1/T_{clustering}$) to balance energy consumption of SNs within a cluster.

6. To save energy, the transmission power of a SN even when it is a CH is reduced to a minimum level to enable the SN to communicate with its neighbor SNs within one-hop radio range denoted by r. Thus, every SN needs to use a multi-hop route (i.e. passing through a number of other SNs) for it to communicate with another SNs distance away. When the WSN becomes less dense as time progresses due to sensor node failures, the one-hop radio range can be increased dynamically to allow the WSN to continue its function at the expense of energy consumption. Also, to save energy, SNs do not operate in the full power mode, but in the power saving mode. At this mode, a SN

operates either in active mode, i.e., transmitting or receiving, or in sleep mode. The radio module of a modern sensor is shown in Figure 2.2 [81], [82]. While in sleep mode, a SN's radio module (shown within the dotted line) is shut off. The analog block which is a part of the receiver is awake and acts as the radio detector. The analog block is shown in Figure 2.3. When the analog block detects a radio signal, the signal is sent through the low noise amplifier (LNA) and converted to a control signal through the DC converter. This control signal is sent to the power control electronics to wake up the radio module. With the state-of-art technology, energy consumed by the analog block is very small. Also the current technology can achieve the transient time between active and sleep mode of 5μs [81]. Therefore, the energy consumed for turning on/off radio while a SN is in power-saving mode is also very small. Thus, we only consider the energy consumed while a SN is transmitting or receiving in active mode.

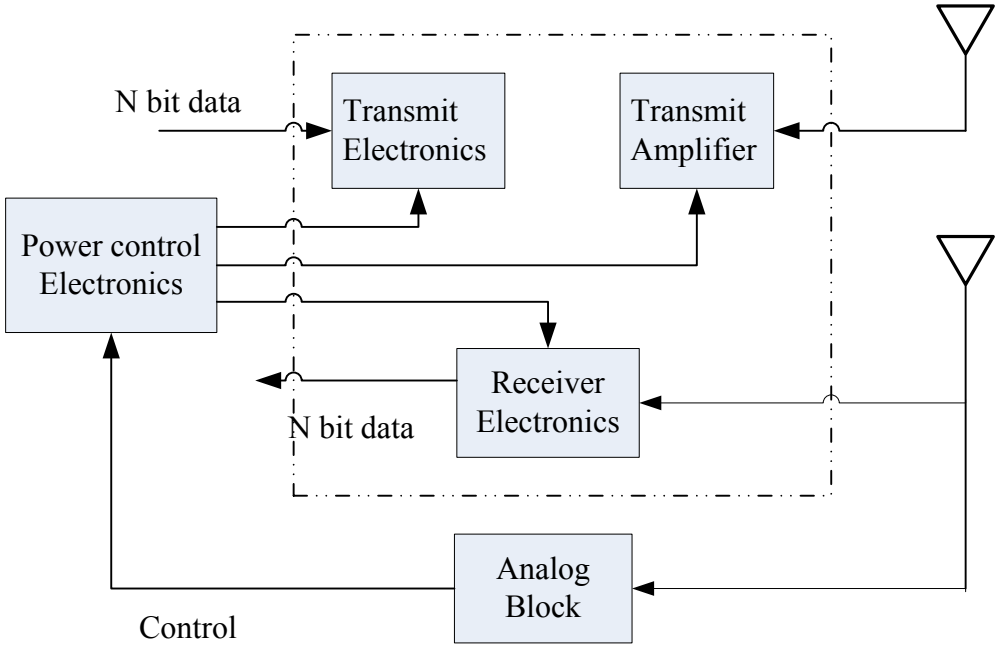

Figure 2-2: Radio Module of a Sensor Transmitter and Receiver.

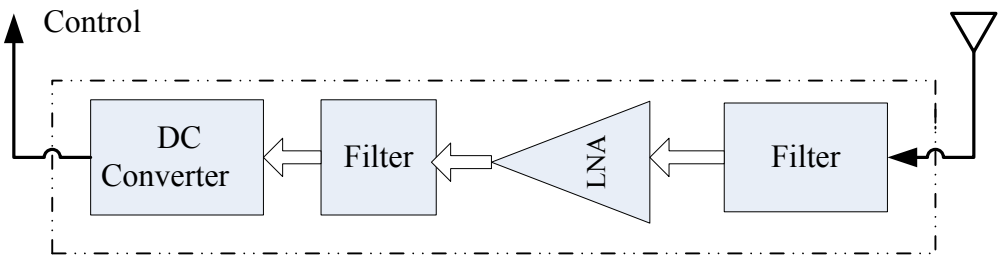

Figure 2-3: Analog Block for Radio Detection.

7. The unreliable transmission failure behavior of the wireless medium in WSNs due to noise and interference is characterized by a transmission failure parameter at hop *j*. This parameter varies among sensors, depending on the node density and the packet transmission rate of SNs within radio range. Let e_j denote the transmission failure probability of SN_j (where $0< e_j <1$). Note that e_j varies dynamically in response to network dynamics.
8. Users (on a flying airplane or a moving vehicle) can issue a query through any CH, which we call it a processing center (PC) or a monitoring node as labeled in Figure 2-1. A query may involve all or a subset of clusters, say, *k* clusters, to respond to the query for data sensing and retrieval. These requested clusters are termed *source clusters*. Multiple queries can be processed concurrently as long as source clusters are different to avoid communication interference. The CH of a source cluster does not aggregate data. It may receive m_s packets carrying the same data content from m_s SN within its cluster because of source redundancy but it will only relay the first return packet to the monitoring center. We assume queries are issued one at a time by the user who is on the move. Thus the timeliness requirement is tight, i.e., on the order of a tenth of a second. The WSN does not have a base station. Also, sensors in a cluster will rotate to be the CH in their cluster. Thus, the notion of higher energy consumption by critical nodes [77] for relaying messages to a base station or to a CH does not exist.
9. A source CH must relay sensor data information to the processing center in response to a user query, and thus can consume more energy than a SN within its cluster. The energy consumed by the system for data forwarding in response to a query depends on the total length (in terms of the number of hops) of the paths connecting m_s SNs within a cluster to the source CH and the total length of the *m* paths connecting the source CH

and the processing center (the destination CH). We assume as a first approximation that the area is relatively free of obstacles and that the WSN is dense enough so that the length of a path connecting two SNs can be approximated by the straight line distance divided by *r*.

10. Routing in the WSN is based on geographic forwarding. No path information needs to be maintained by individual SNs to conserve energy. Essentially only the location information of the destination SN needs to be known by a forwarding SN for any source-destination communication. When a CH is elected periodically, the location information is broadcast to the WSN to let other CHs know its location. SNs within a cluster will know the location of their CH as part of the CH election process.
11. As the clustering algorithm in effect rotates SNs within a cluster equally to assume the role of the CH, each SN would consume energy at about the same rate. Thus, instead of considering each individual sensor energy level, we can consider the system energy whose initial energy level is given by $E_{initial} = nE_o$, where E_o is the initial energy of a sensor node. When the energy level of the system falls below a threshold value, say $E_{threshold}$, the WSN is considered as having depleted its energy. For the energy model, we use the general radio transmission model adopted in many recent works [60], [63], [79] - [88]. In this model, we assume all SNs use the same radio range denoted by *r*. The SNs work in power saving mode as stated in assumption 6. In the power saving mode, a node is active and performs its functions for a fraction of time. It sleeps to save energy most of the time. In sleep mode, a node stops its communication functions, but may perform other activities, such as sensing and signal detecting. In active mode, SNs can transmit and receive packets. The transmission power is related mainly to radio range and other parameters such as bit error rate (BER), modulation, bandwidth and frequency. The path loss depends on the radio range following the law of exponents, where the path lost exponent α is usually between 2 and 6 in different environments. The energy dissipations in each node at sleeping, receiving and transmitting states are denoted by E_S, E_R, and E_T respectively. The sleep energy E_S is usually small compared with E_R and E_T. The energy expended to receive a message of length n_b bit is given by:

$$E_R = n_b E_{Rx-elec} \qquad (1)$$

where $E_{Rx\text{-}elec}$ is the reception cost to run the radio circuitry per bit processed (J/bit).

The energy spent by a SN to transmit a data packet of length n_b bits for a distance of r is given by:

$$E_T = n_b \left(E_{Tx-elec} + \varepsilon_{amp} r^{\alpha} \right) \qquad (2)$$

where $E_{Tx\text{-}elec}$ is the transmission cost to run the radio circuitry per bit processed (J/bit). The energy cost of the transmit amplifier denoted by ε_{amp} is to achieve an acceptable signal to noise ratio. The path loss exponent α depends on the transmission conditions. For our model we choose α = 2 since the area is assumed to be flat and the environment is considered as free space. The parameter ε_{amp} in this case is scaled by J/bit/m^2 and usually determined by experiment.

We define the *mean time to failure* (MTTF) of a query-based WSN as the total number of queries the system can answer correctly until it fails due to channel or sensor faults, or when the system energy reaches the energy threshold level $E_{threshold}$. We define a query's QoS requirements in terms of its *reliability* and *timeliness* requirements, denoted as R_{req} and T_{req}. For example, a patrol, search and rescue vehicle (PSAR) application usually has a very tight deadline requirement. Energy is not a query QoS requirement. The objective of the design is to prolong the system lifetime; therefore the energy consumption must be reflected in the calculation of MTTF. The system must deliver query results within T_{req} and the reliability of data delivery must be at least R_{req}. Our objective is to determine the best path and source redundancy levels to satisfy QoS while maximizing MTTF. When the frequency of queries is known, the MTTF parameter can be translated into the conventional system *lifetime* parameter.

Chapter 3
ADAPTIVE FAULT TOLERANT QOS CONTROL ALGORITHM

In this chapter, we describe the proposed adaptive fault tolerant QoS control (AFTQC) algorithm and its implementation using two different approaches, namely, proactive and reactive. We also describe how we extend the AFTQC algorithm to handle concurrent queries and adapt to network dynamics due to energy consumption and failure of sensor nodes in WSNs.

3.1. Design Strategies

Our design of fault tolerant QoS control algorithms centers on the concept of tolerating sensor faults and communication faults due to noise and interference. We consider two mechanisms for implementing fault tolerant QoS control algorithms, namely, source redundancy and path redundancy. The source redundancy mechanism offers source-level fault tolerance ranging from no redundancy, dual module redundancy with fault detection, to triple or multiple module redundancy with fault masking to return a sensor reading in a feature area. The path redundancy mechanism offers routing path fault tolerance ranging from one path (no redundancy) to multiple disjoint/braided paths for a sink-source pair. The AFTQC algorithm selects m_s SNs for source redundancy within a cluster to return a sensor reading to their cluster head, and m paths for path redundancy between the source cluster head and the processing center to forward sensing results, with the goal to satisfy query QoS requirements while maximizing the lifetime of WSNs. A dynamic AFTQC algorithm selects the optimal (m, m_s) dynamically in response to network dynamics such as failures of nodes, density of nodes, energy of nodes, and node connectivity.

3.2. Hop-by-Hop Data Delivery Protocol

To minimize energy consumption, our algorithms do not use routing tables to maintain routes. Rather we leverage geographical routing that allows SNs to route information hop-by-hop to their CH and then from the CH to the processing center node. We develop a *hop-by-hop data*

delivery (HHDD) protocol to implement the desired level of redundancy to achieve QoS. For path redundancy, we want to form m paths from a source CH to the processing center, as illustrated in Figure 3-1. This is achieved by having m SNs in hop one relay the data through broadcasting, and only one single SN in all subsequent hops relay the data per receiving group. For source redundancy, we want each of the m_s SNs to communicate with the source CH through a distinct path. This is achieved by having only one SN relay the data through broadcast in each of the subsequent hops in each path. Certainly, a WSN is inherently broadcast based. However, a SN can specify a set of SNs in the next hop (that is, m in the first hop and 1 in a subsequent hop) as the intended receivers and only those SNs can forward data.

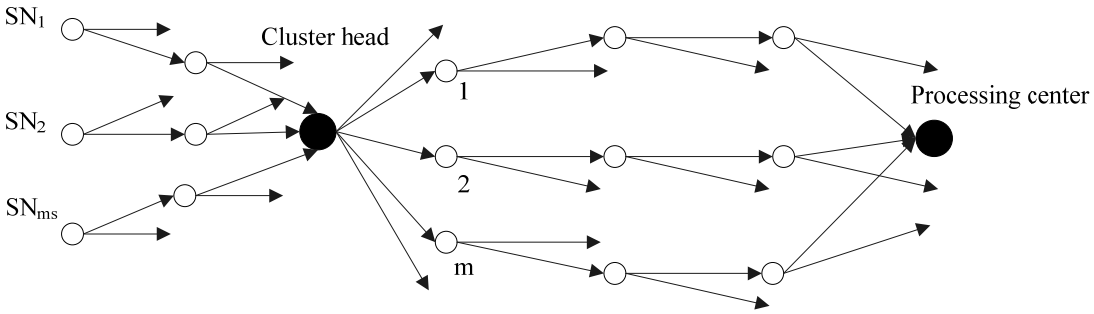

Figure 3-1: Hop-by-Hop Data Delivery (HHDD).

3.3. AFTQC Algorithm Description

Here we describe our *adaptive fault tolerant QoS control (AFTQC)* algorithm. The algorithm determines the best redundancy level to be used for data propagation, based on the current channel/node and transmission delay conditions, with the goal to satisfy QoS requirements and to maximize system lifetime of a WSN. The algorithm is also designed to adapt to network dynamics due to changes in sensor node density (λ), sensor node residual energy (E_0), and transmission radius (r) as time progresses.

Our AFTQC algorithm works as follows. A cluster head will collect information at runtime on a per query basis to parameterize the following two parameters: transmission failure probability of node SN_j (e_j) and transmission speed violation probability ($Q_{t,jk}$) between SN_j and SN_k. Other system parameters such as the total number of sensors (n), the number of sensors in a cluster (n_s), transmission radius (r), and sensor density (λ) will be collected periodically. Then,

by performing a lookup operation into the best (m, m_s) table, called the MMS table, built at design time, a processing center will determine the best (m, m_s) that would maximize the MTTF. The source CH then will implement the best (m, m_s) by following the HHDD protocol. The MMS table built at design time can be obtained by calculating the MTTF as a function of (m, m_s) based on an analytical model developed in this dissertation in Chapter 4, covering a perceivable set of baseline environment parameter values (e_j, $Q_{t,jk}$) and listing the best (m, m_s) that can maximize the MTTF. Our algorithm is adaptive to network conditions as it collects and parameterizes model parameters and can dynamically determines the best (m, m_s) to maximize the MTTF while satisfying the application QoS requirements. Further, it rebuilds the MMS table in response to network dynamics. In the following, we discuss two ways of collecting values of model parameters, namely, proactive vs. reactive, at runtime for the implementation of the algorithm.

3.4. Proactive vs. Reactive AFTQC Algorithm

In this section, we describe two approaches of implementing the AFTQC algorithm. Section 3.4.1 describes the proactive AFTQC algorithm. Section 3.4.2 describes the reactive AFTQC algorithm. For each approach we describe how status reporting and query processing are performed. We also discuss the pros and cons of these approaches in section 3.4.3.

3.4.1. Proactive AFTQC

The proactive AFTQC algorithm is illustrated in Figure 3.2.

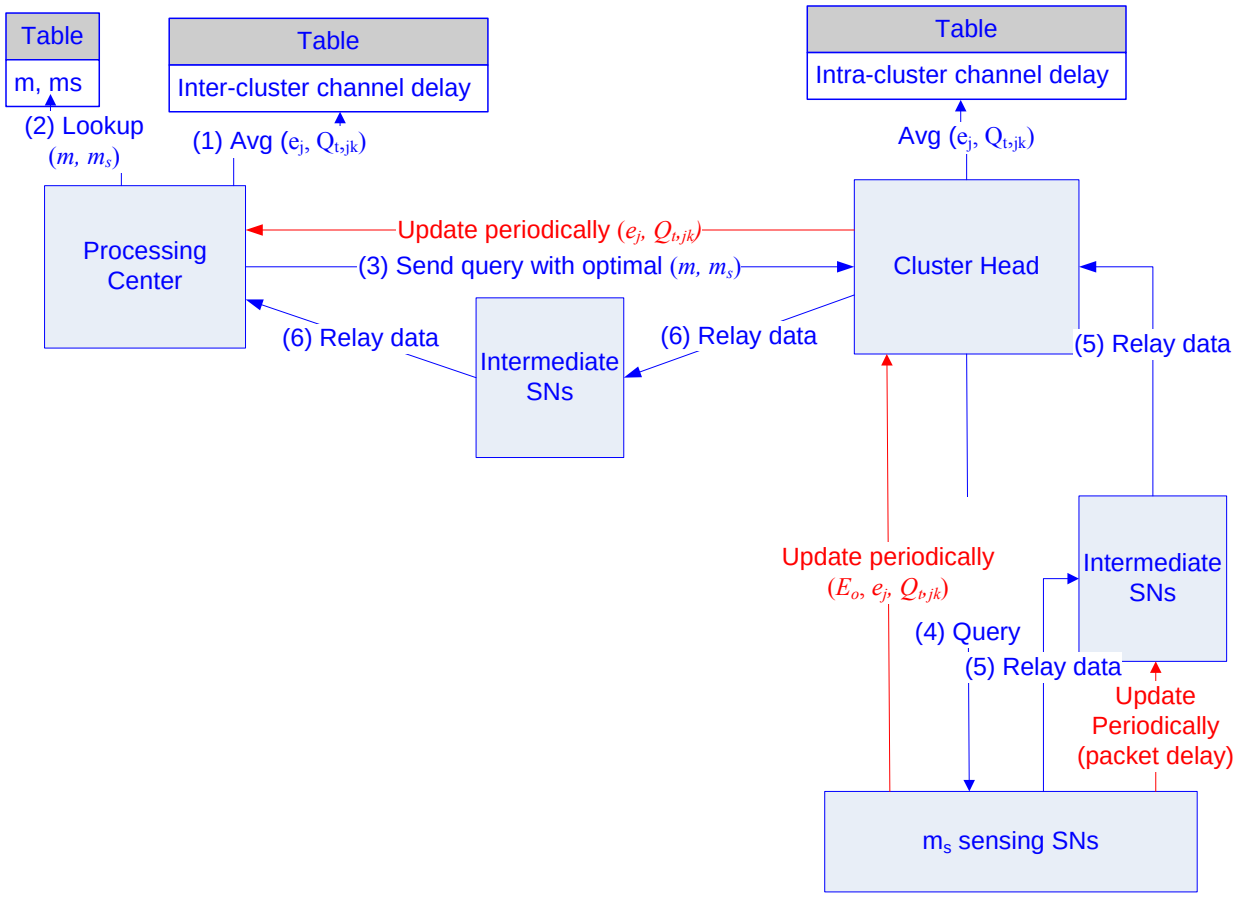

Figure 3-2: Proactive AFTQC Algorithm.

A. Status Reporting

Under the proactive approach, a SN, say, SN_j, periodically exchanges its location and packet delay information with its one-hop neighbors SN_k. This allows SN_j to calculate the progressive speed (S_{jk}) between SN_j and SN_k, and also the transmission speed violation probability ($Q_{t,jk}$) between SN_j and SN_k. Periodically, SN_j also sends the status update on (e_j, $Q_{t,jk}$) to its CH. A CH will calculate the average (e_j, $Q_{t,jk}$) of all the SNs in its cluster and store this information in an *intra-cluster channel/delay* table. Further, CHs also periodically exchange the average (e_j, $Q_{t,jk}$) information with other CHs and store the information in an *inter-cluster channel/delay* table. When a SN is assigned to perform sensing, it will perform the sensing task, relay the sensor reading, and then go back to the power-saving mode. Similarly, SNs that are assigned to forward packets will go back to the power-saving mode after packets are forwarded.

SNs also periodically (after each T period) send a status packet to their CH to inform the CH of their remaining energy level. The CH will store this information in the intra-cluster table. When a cluster is selected to answer a query, the CH uses the energy information kept in the intra-cluster table to select m_s SNs to answer the query.

B. Query Processing

Step 1: Determine the optimal level of redundancy (m, m_s) and send the query: The processing center (PC) will first determine the optimal level of redundancy (m, m_s) to answer a query to satisfy the required reliability (R_q) and timeliness (T_q) requirements. The PC will send a packet to the source CH carrying the optimal redundancy (m, m_s). To determine the optimal (m, m_s), the PC needs to know the average values of the transmission failure probability (e_j) and transmission speed violation probability ($Q_{t,jk}$) at runtime.

Each CH periodically (after each T period) sends a packet to other CHs carrying the information of average (e_j, $Q_{t,jk}$). This information will be stored in the *inter-cluster channel/delay* table. When a query arriving at cluster A (thus the CH of cluster A is the PC) demands responses from a source cluster, say, cluster B, a table lookup into its *inter-cluster channel/delay* table by the PC is performed to retrieve the average (e_j, $Q_{t,jk}$) value out of those CHs located between clusters A and B. These average (e_j, $Q_{t,jk}$) values then can be used as indexes into the MMS table to lookup for the optimal level of redundancy (m, m_s) that should be used in response to the query. The PC then sends (m, m_s) along with the query packet to the source CH.

Step 2: Chose m_s sensors to respond to the query: In this step, the source CH chooses m_s SNs that should respond to the query and send them a command packet carrying the query. The command packet is sent to m_s sensors reliably by unicasting. The CH chooses m_s SNs that have the highest remaining energy level to execute the query.

Step 3: Relay sensor data from SNs to CH: In this step, the m_s chosen SNs will perform sensor reading and forward sensor data to their CHs using the HHDD protocol. A SN chooses the first next-hop SN to forward data if the progressive speed between these two nodes satisfies the speed requirement (S_{req}) so that only one path would be formed between the CH and a chosen SN. Each intermediate SN also discards duplicate packets.

Step 4: Relay sensor data from the CH to the PC: In this step, the CH will relay data to the PC using the HHDD protocol. The CH broadcasts a data packet carrying the query reply to the first-hop SNs but specifies in the packet m SNs (if exist) with progressive speeds S_{jk} satisfying the speed requirement (S_{req}) to forward the packet. Recall that each SN periodically exchanges its location and packet delay information with its one-hop neighbors, so each SN knows which of its one-hop neighbors will satisfy the speed requirement. Each intermediate SN selected to forward data in turn specifies in the data packet exactly one of the next-hop SNs with the progressive speed satisfying the speed requirement to continue forwarding data. This way, we form m paths between the CH and the PC to achieve path-level fault tolerance.

In summary, the proactive AFTQC is described below in pseudo code:

Processing Center (PC):

1. Performs a table lookup operation into its *inter-cluster channel/delay* table to determine the average (e_j, $Q_{t,jk}$) of the CHs between the source CH and the PC.
2. Performs table lookup into MMS table using the average (e_j, $Q_{t,jk}$) values to determine the optimal level of redundancy (m, m_s).
3. Sends the query and the optimal redundancy level (m, m_s) to the source CH.

Cluster Head (CH):

1. After each T period, sends an update on the average (e_j, $Q_{t,jk}$) to other CHs.
2. When receiving a query from the PC to serve as the source CH, chooses m_s SNs to perform data sensing.
3. When receiving query replies from m_s sensors, relays the result to PC by following the HHDD protocol.
4. Stores the residual energy information received from SNs within the cluster in the intra-cluster table.
5. Computes the average (e_j, $Q_{t,jk}$) based on (e_j, $Q_{t,jk}$) received periodically from SNs within the cluster and store the average (e_j, $Q_{t,jk}$) in the intra-cluster channel/delay table.
6. Stores the average (e_j, $Q_{t,jk}$) received from other CHs in the inter-cluster channel/delay table.

Sensor Node (SN):

1. After each T period, exchanges location and packet delay information with its one-hop neighbors and sends a status packet to CH to inform of its residual energy (E_o) and the updated (e_j, $Q_{t,jk}$) parameters.
2. When receiving a query command from the CH, performs sensor reading.
3. Sends data to the CH based on the HHDD protocol.

Intermediate Node (SN_I):

1. When receiving a data broadcast message, checks to see if it is specified as a forwarding node. If yes, rebroadcasts the data packet but specifies in the data packet only one SN that satisfies the speed requirement in the next hop to continue forwarding the data.
2. Discards duplicate data.

3.4.2. Reactive AFTQC

The reactive AFTQC is illustrated in Figure 3.3.

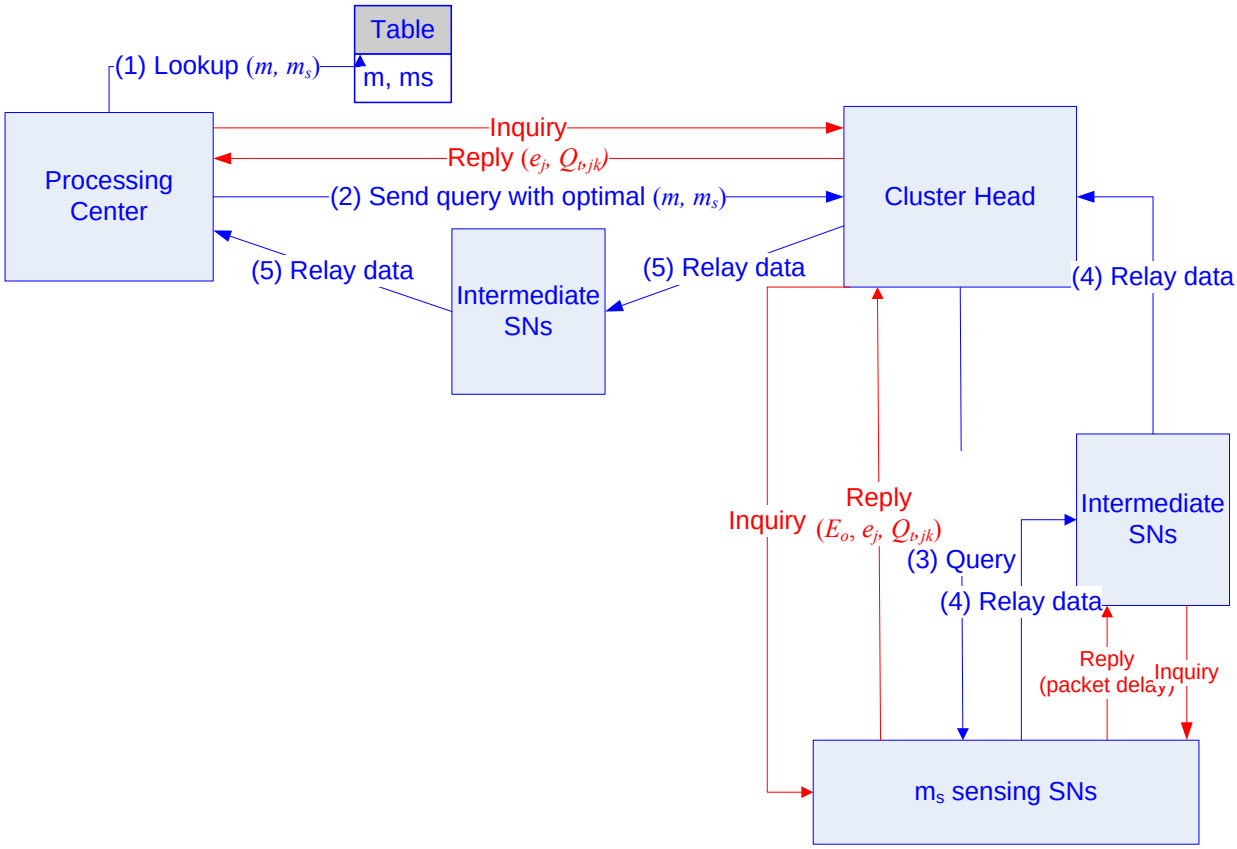

Figure 3-3: Reactive AFTQC Algorithm.

A. Status Reporting

All SNs will be in power-saving mode and awake upon receiving an inquiry. For a status update packet sent by a CH, a SN will wake up to (1) send a request to its one-hop neighbors for the information on packet delay information, (2) receive the reply from its one-hop neighbors and calculate the transmission speed violation probability $Q_{t,jk}$, (3) send the reply to the CH, and (4) go back to the power-saving mode. When SNs receive a query to perform sensing, they will perform the sensing task, relay the sensor reading, and then go back to the power-saving mode. Similarly, SNs that are assigned to forward packets will go back to the power-saving mode after packets are forwarded. For a status update request sent by the PC, a CH will send back the reply on the average (e_j, $Q_{t,jk}$) value.

B. Query Processing

Step 1: Determine the optimal level of redundancy (m, m_s) and send the query: The PC will send an inquiry packet to the source CH as well as all CHs located between the PC and the source CH to request an update on the average (e_j, $Q_{t,jk}$) of SNs in their clusters. Upon receiving the request from the PC, CHs will send back a reply packet carrying the average (e_j, $Q_{t,jk}$) to the PC. These average (e_j, $Q_{t,jk}$) values then can be used as indexes into the MMS table to lookup for the optimal level of redundancy (m, m_s) that should be used in respond to the query. The PC then sends (m, m_s) along with the query packet to the source CH.

All other steps, including Step 2 (Choose m_s sensors that should respond to the query), Step 3 (Relay sensor data from SNs to CH), and Step 4 (Relay sensor data from the CH to the PC) are the same as listed in the proactive approach.

In summary, the reactive AFTQC approach is described below in pseudo code:

Processing Center (PC):

1. Sends an inquiry packet to the source CH and those CHs between the source CH and the PC to request information on (e_j, $Q_{t,jk}$).
2. Performs table lookup into the MMS table using the average (e_j, $Q_{t,jk}$) values to determine the optimal level of redundancy (m and m_s).
3. Sends the query and the optimal redundancy level (m, m_s) to the source CH.

Cluster Head (CH):

1. When receiving an inquiry form the PC or other CHs between the PC and the source CH, sends an update on (e_j, $Q_{t,jk}$) to the PC or other CHs.
2. When receiving a query from the PC to serve as the source CH, chooses m_s SNs to perform data sensing.
3. Relays the sensor reading to the PC by broadcasting the data packet to m first-hop neighbor nodes.

Sensor Node (SN):

1. When receiving an inquiry packet from a neighbor SN, sends packet delay information to the neighbor.
2. When receiving an inquiry packet from the CH, sends a status packet to the CH to inform of its residual energy (E_o) and (e_j, $Q_{t,jk}$).
3. When receiving a query from the CH, performs sensor reading.
4. Sends data to the CH based on the HHDD protocol.

Intermediate Node (SN_I):

1. When receiving a data broadcast message, checks to see if it is specified as a forwarding node. If yes, rebroadcasts the data packet but specifies in the data packet only one SN that satisfies the speed requirement in the next hop to continue forwarding the data.
2. Discards duplicate data.

3.4.3. Strengths and Weaknesses

Here we discuss the pros and cons of proactive vs. reactive approaches. For the proactive approach, when the periodic update interval is short, it may incur more energy consumption at the CHs and SNs for status exchanges. Extra energy consumption also incurs at the source CH to maintain the inter-cluster channel/delay and intra-cluster channel/delay tables. When a new SN becomes the CH due to CH rotation, these tables may need to be transferred to the new CH, especially if the status exchange period is long. This process will also incur overhead and consume extra energy. For the reactive approach, when the query arrival rate is high, it may incur more energy used at the CHs and SNs to send requests and responses for status update on (e_j, $Q_{t,jk}$) and the residual energy level of SNs. The dissertation research analyzes these approaches quantitatively by means of mathematical modeling and analysis in Chapters 4 and 5, and verifies them by simulation in Chapter 6.

Chapter 4
PROBABILITY MODEL AND ANALYSIS

In this chapter, we present probability models to analyze our adaptive fault tolerant QoS control algorithm. To process a query, our algorithm deals with two parts: 1) forward traffic and 2) reverse traffic. In the forward traffic, the query is distributed from the PC to the source CH, and then from the source CH to selected SNs within the cluster. In the reverse traffic, the responses are relayed from SNs to the source CH and then from the source CH to the PC.

In Section 4.1 we derive analytical expressions for the reliability and energy consumption for processing a query in both the forward traffic and the reverse traffic. Other than query processing, three other activities in the system also consume energy and affect the lifetime of the WSN. The first is the periodic clustering activity for clustering nodes. This is analyzed in Section 4.2. The second is the status exchange activity for nodes to exchange E_o, e_j, Qt_{jk}, which can be done proactively or reactively. This is described in Section 4.3. The third is the rebuilt MMS table activity in response to network dynamics. This is discussed in Section 4.5.

After we derive analytical expressions for the reliability and energy consumption for these activities, in Section 4.4 we derive MTTF of the WSN under our design. In Sections 4.5-4.8 we discuss extensions to the basic design. In Section 4.5 we analyze energy consumption for periodically rebuilding the MMS table for determining the best (m, m_s) to use dynamically. In Section 4.6, we analyze network dynamics and propose solutions for the system to dynamically determine network parameter values to reflect changes in the sensor network environment. In Section 4.7, we generalize the model to relax some of assumptions made. This includes how we model and analyze concurrent query processing, queries with distinct QoS requirements or involving multiple clusters for a response, and WSNs with data aggregation functionality. Finally in Section 4.8, we extend the model to consider the case in which *acknowledgement (ACK) with timeout* is being used in the design and explore the tradeoff in reliability vs. energy consumption compared with the no-ACK design. In particular, the ACK with timeout and no redundancy design is used as a baseline model, against which our design is compared to demonstrate the feasibility and benefit of our design.

Two forms of redundancy are considered for which we analyze their effect on the lifetime of the WSN. The first one is path redundancy. That is, instead of using a single path to connect a source cluster to the processing center, m disjoint paths may be used. The second is source redundancy. That is, instead of having one SN in a source cluster return requested sensor data, m_s SNs may be used to return readings to cope with data transmission and/or sensor hardware faults. Figure 2-1 illustrates the case in which m=2 and m_s =5.

The objective of the probability model is to express the MTTF metric as a function of model parameters including m, m_s, e_j, $Q_{t,jk}$, λ, E_o, r, where m, and m_s are the redundancy level to be determined that would maximize the MTTF, while e_j, and $Q_{t,jk}$ are to be estimated by a PC at runtime on a query by query basis through status exchange among CHs, and λ, E_o, r are to be updated on a periodic basis to reflect network dynamics due to node failures, change of energy, change of node density and radio range. The way we calculate MTTF is to find out the maximum number of queries the system can satisfy before running out of energy, assuming every query has a reliability of 1. Then we consider the reliability of the query to find out the average number of queries the system can sustain. The closed-form solution derived allows the MTTF metric to be computed. Ideally, a MMS table would be built at design time listing the MTTF as a function of m, m_s, e_j, $Q_{t,jk}$, λ, E_o, r. When the SN storage is an issue, the table may list MTTF as a function of m, m_s, e_j, $Q_{t,jk}$, and the table is rebuilt periodically with λ, E_o, and r being updated periodically

4.1. Reliability and Energy Consumption of Query Processing

Let R_q be the reliability of a query as a result of applying our proposed hop-by-hop data delivery protocol with m paths for path level redundancy and m_s sensors for source level redundancy. Let E_q be the average energy consumption of the system to answer a query. For simplicity, assume only one source cluster is needed to answer a query. Let R_q^R and R_q^F be the query reliabilities of the reverse traffic and the forward traffic, respectively. Then, the overall query success probability is given by:

$$R_q = R_q^F R_q^R \qquad (3)$$

Let E_q^R and E_q^F be the energy consumed by the reverse traffic and the forward traffic, respectively. Then, the overall energy consumption for query processing (per query) is given by:

$$E_q = E_q^F + E_q^R \qquad (4)$$

Below in Section 4.1.1 we derive analytical expressions for R_q^R and R_q^F and in Section 4.1.2 we derive analytical expressions for E_q^R and E_q^F.

4.1.1. Query Processing Reliability

Here we will first derive R_q^R and then R_q^F. Let d_{inter} be a random variable denoting the distance between a source CH and the processing center and let d_{intra} be a random variable denoting the distance between a SN to the CH. Then, the number of hops (excluding the processing center) between the processing center to the source CH, denoted by h, is given by:

$$h = \left\lceil \frac{d_{inter}}{r} \right\rceil \qquad (5)$$

A query can be initiated by any CH which serves as the processing center for that query. Thus, the location of the processing center varies on a query by query basis. For derivation convenience without loss of generality, let the processing center be located in the center of the sensor area with the coordinate at (0, 0) and the source CH be randomly located at (X_i, Y_i) in the rectangular sensor area with $-A/2 \le X_i \le A/2$ and $-B/2 \le Y_i \le B/2$. Then, the expected value of d_{inter} is given by:

$$E[d_{inter}] = \left(\frac{1}{AB}\right) \int_{-A/2}^{A/2} \int_{-B/2}^{B/2} \sqrt{(X_i^2 + Y_i^2)}) dX_i dY_i \qquad (6)$$

For the case of square area, i.e. $A = B$, we obtain $E[d_{inter}] = 0.3825$ A

The same final expression for $E[d_{inter}]$ would result if we had taken the coordinate of the processing center to be (X_c, Y_c) in the square sensor area and put two more integrals, one for X_c and the other for Y_c with $-A/2 \le X_c \le A/2$ and $-A/2 \le Y_c \le A/2$, because of symmetric properties. For notational convenience, let N_{inter}^h represent the average number of hops (or sensors) to forward sensor data from a source CH to the processing center.

$$N_{inter}^{h} = \lceil E[h] \rceil = \left\lceil \frac{0.3825A}{r} \right\rceil \qquad (7)$$

Since a sensor becomes a CH with probability *p* and all the sensors are distributed in the area in accordance with a spatial Poisson process with intensity λ, the CH and non-CH sensors will also be distributed in accordance with a spatial Poisson process with rates pλ and (1-p)λ, respectively. Non-cluster-head sensors thus would join the cluster of the closest CH to form a Voronoi cell [76] corresponding to a cluster in the WSN. It has been shown that [47], [59] the average number of non-cluster-head sensors in each Voronoi cell is (1-p)/p and the expected distance from a non-cluster-head sensor to the CH is given by:

$$E[d_{intra}] = \frac{1}{2(p\lambda)^{1/2}} \qquad (8)$$

If this distance is more than per-hop distance *r*, a sensor will take a multi-hop route to transmit sensor data to the CH. The average number of intermediate sensors (including the sensor itself) is the quantity above divided by per-hop distance *r*. Let N_{intra}^{h} denote the average number of hops to forward sensor data from a SN responsible for a reading to its CH. Then, N_{intra}^{h} is given by:

$$N_{intra}^{h} = \left\lceil \frac{1}{2r(p\lambda)^{1/2}} \right\rceil \qquad (9)$$

Let $Q_{r,j}$ be the probability of a SN, say, SN_j, failing to relay sensor data because of either a sensor failure or a transmission failure, or both. Then $Q_{r,j}$ is given by:

$$Q_{r,j} = 1 - [(1-q)(1-e_j)] \qquad (10)$$

Let the deadline requirement of a query be T_{req} and the time spent for the forward traffic be $Time^F$ and the time spent for the reverse traffic be $Time^R$. Since the path length in the forward traffic and in the reverse traffic is about the same, the difference between $Time^F$ and $Time^R$ is caused by the traffic load on the forward or reverse traffic. Thus,

$$\frac{Time^R}{Time^F} = \frac{n_b}{n_q} \qquad (11)$$

where n_b is the average size of a data packet and n_q is the average size of a query. In addition to the time spent for the forward and reverse traffic, we also need time to probe node status for reactive AFTQC. Let the time spent for status exchange be $Time^{status}$ for reactive AFTQC (it is zero for proactive AFTQC). In order to satisfy the deadline requirement, we have the following constraint:

$$Time^R + Time^F + Time^{status} = T_{req} \qquad (12)$$

Therefore, the time spent for the forward traffic is constrained by:

$$Time^F = (T_{req} - Time^{status}) \frac{n_q}{(n_b + n_q)} \qquad (13)$$

The time spent for the reversed traffic is constrained by:

$$Time^R = (T_{req} - Time^{status}) \frac{n_b}{(n_b + n_q)} \qquad (14)$$

To estimate $Time^{status}$ for reactive AFTQC we observe that status exchange between the PC and the CHs can be done simultaneously. Similarly, status exchange can be done simultaneously between the source CH and SNs in a cluster, as well as between a SN and its one-hop neighbors. Let n_{st} be the status exchange packet size. Therefore, the time taken for the PC to send requests to the CHs in between the PC and the source CH (including the source CH) and receive replies from them can be calculated by $2(N^h_{\text{int}er} n_{st} / B)$ where $N^h_{\text{int}er}$ is determined by Equation (7). Note that here we take the longest distance between the PC and the source CH as the bottleneck. Similarly the time taken for the CH to send requests to the SNs in its cluster and receive replies from them is calculated by $2(N^h_{\text{int}ra} n_{st} / B)$ where $N^h_{\text{int}ra}$ is determined by Equation (8). The time taken for a SN to send the request and receive a reply from its one-hop neighbors is

calculated by $2n_{st}/B$. Summarizing above, the total time for status exchange for reactive AFTQC is calculated by $2\frac{n_{st}}{B}(1+N^{h}_{\mathrm{int}er}+N^{h}_{\mathrm{int}ra})$.

Since the distance from the PC to the SNs performing sensing is given by d_{inter}+ d_{intra}, the per-hop minimum *speed* requirement, to satisfy the timing constraint for the forward traffic is given by:

$$S^{F}_{req}=\frac{d_{inter}+d_{intra}}{Time^{F}} \tag{15}$$

The per-hop minimum *speed* requirement to satisfy the timing constraint for the reversed traffic is given by:

$$S^{R}_{req}=\frac{d_{inter}+d_{intra}}{Time^{R}} \tag{16}$$

Plugging in the expected values of d_{inter} and d_{intra}, the expected *speed* requirement for the forward traffic is given by:

$$E[S^{F}_{req}]=\frac{0.3825A+\frac{1}{2(p\lambda)^{1/2}}}{Time^{F}} \tag{17}$$

The expected *speed* requirement for the reversed traffic is given by:

$$E[S^{R}_{req}]=\frac{0.3825A+\frac{1}{2(p\lambda)^{1/2}}}{Time^{R}} \tag{18}$$

Let $Q_{t,jk}$ be the probability that the speed requirement is violated when a packet is forwarded to SN_k from SN_j. To calculate $Q_{t,jk}$ we need to know the speed S_{jk} from SN_j to SN_k. This can be dynamically measured by SN_j following the approach described in [48], [51]. The progressive speed S_{jk} is calculated by dividing the advance in distance from the next hop node SN_k by the estimated delay (including queueing, processing, and MAC collision resolution) to forward a packet to node SN_k, i.e. $S_{j,k}=(dist_{j,d}-dist_{k,d})/delay_{j,k}$. If S_{jk} is above $E[S_{req}]$ then

$Q_{t,jk}$ = 0; otherwise, $Q_{t,jk}$ = 1, where $S_{req} = E[S^F_{req}]$ for the forward traffic and $S_{req} = E[S^R_{req}]$ for the reverse traffic. In general S_{jk} is not known until runtime. If S_{jk} is uniformly distributed within a range [*a, b*], then $Q_{t,jk}$ can be computed as:

$$Q_{t,jk} = cdf(S_{jk} \le E[S_{req}]) = \frac{E[S_{req}] - a}{b - a} \qquad (19)$$

Let n_k be the average number of one-hop neighbors, calculated as $\lambda\pi r^2$. It has been reported that the number of edge-disjoint paths between nodes is equal to the average node degree with a very high probability [49]. Thus when the density is sufficiently high such that n_k is sufficiently larger than m and m_s, this hop-by-hop data delivery scheme can effectively result in m redundant paths for path redundancy and m_s distinct paths from m_s sensors for source redundancy.

The probability of SN_j failing to relay a broadcast packet to a one-hop neighbor SN_k because of either sensor/channel failures, or speed violation, denoted by $Q_{rt,jk}$, is given by:

$$Q_{rt,jk} = 1 - [(1 - Q_{r,j})(1 - Q_{t,jk})] \qquad (20)$$

The probability that *at least* one next-hop SN (among the one-hop neighbors) of SN_j along the direction of the destination node is able to satisfy the speed requirement and receive the broadcast message is given by:

$$\theta_j = 1 - \prod_{k=1}^{f \times n_k} Q_{rt,jk} \qquad (21)$$

Here n_k is the number of neighbors; f is the fraction of neighbors that would forward data based on geographical routing, e.g., f=1/4 meaning only the sensors along the quadrant toward the direction of the target node will do data forwarding. Note that while SN_j forwards data to its one-quadrant neighbors SN_k's, if one of SN_k's is the destination node, then the probability that the destination SN fails to receive the message due to sensor/channel failures or speed violation is exactly equal to $Q_{rt,jk}$ as given in Equation (20). Below we derive the probability that a path is successfully formed for hop-by-hop data delivery between the source CH and the processing center. Since there are N^h_{inter} hops between the source CH (the first SN with index 1), and the

processing center (the last SN with index N^h_{inter}+1), a path is formed for data delivery if in each hop there is *at least* one next-hop sensor along the direction of the target node is able to satisfy the speed requirement and receive the broadcast message, *and* also that the destination node is able to satisfy the speed requirement and receive the message. Thus, the probably that a path of length N^h_{inter} is formed successfully for hop-by-hop data delivery is given by:

$$\Theta(N^h_{\text{int}er}) = (\prod_{j=1}^{N^h_{\text{int}er}-1} \theta_j) \times (1 - Q_{rt, N^h_{\text{int}er}(N^h_{\text{int}er}+1)}) \tag{22}$$

where $Q_{rt,\, N^h_{inter}(N^h_{inter}+1)}$ is from Equation 20 for the probability that the processing center node (the last SN with index $N^h_{inter}+1$) fails to receive the message due to sensor/channel failures or speed violation. Note that the product term in Equation (22) becomes 1 when the upper bound is smaller than the lower bound.

We create *m* paths between the source CH and the processing center based on the hop-by-hop data delivery scheme discussed earlier. This applies to the both the forward traffic and reverse traffic. For the reverse traffic, the source cluster will fail to deliver data to the processing center if one of the following happens:

1. None of the SNs in the first hop receives the message. The probability for this case is $1-\theta_1$.
2. In the first hop, i $(1 \le i < m)$ SNs receives the message, and each of them attempts to form a path for data delivery. However, all i paths fail to deliver the message because the subsequent hops fail to receive the broadcast message. The failure probability for this case is:

$$\sum_{|I|<m} \{[\prod_{i \in I} (1 - Q_{rt,1i})](\prod_{i \notin I} Q_{rt,1i})\} \{\prod_{i \in I} [1 - \Theta_i(N^h_{\text{int}er} - 1)]\}$$

where I stands for a set consisting of first-hop SNs that receive the message and $|I|$ is the cardinality of set I. The first term is the probability that i SNs from the set of $f\, n_k$ nodes in the first hop successfully receive the message, and the second term is the probability that all i paths fail to deliver data. Note that a subscript i has been used to label Θ_i to refer to path i (i.e., the path that starts from a particular SN with index i). Also the argument to Θ_i is only $N^h_{inter}-1$ because there is one less hop to be considered in each path.

3. In the first hop, at least m SNs receive the broadcast message from the source CH, but all m paths fail to deliver the message because the subsequent hops fail to receive the broadcast message. The probability for this case is:

$$\sum_{|I|\geq m}\{[\prod_{i\in I}(1-Q_{rt,1i})](\prod_{i\notin I}Q_{rt,1i})\}\{\prod_{\substack{i\in M,\\ M\subseteq I,\\ |M|=m}}[1-\Theta_i(N^h_{\text{int}\,er}-1)]\}$$

where M is a subset of I with cardinality of m. The second term in the above expression is the probability that all m paths fail to delivery data.

Thus, the probability of the source cluster failing to deliver data to the processing center is given by:

$$\begin{aligned} Q^m_{fp} &= 1-\theta_1 + \\ &\sum_{|I|<m}\{[\prod_{i\in I}(1-Q_{rt,1i})](\prod_{i\notin I}Q_{rt,1i})\}\{\prod_{i\in I}[1-\Theta_i(N^h_{\text{int}\,er}-1)]\} \\ &\sum_{|I|\geq m}\{[\prod_{i\in I}(1-Q_{rt,1i})](\prod_{i\notin I}Q_{rt,1i})\}\{\prod_{\substack{i\in M,\\ M\subseteq I,\\ |M|=m}}[1-\Theta_i(N^h_{\text{int}\,er}-1)]\} \end{aligned} \tag{23}$$

For source redundancy, instead of using one SN, we assign m_s SNs in each cluster to return sensor readings to their CH to cope with channel/sensor faults. To implement source redundancy, SNs also use hop-by-hop data delivery based on geographical routing to send sensor data to their CH. For a path of N^h_{intra} from a SN to the CH, again assign an index of 1 to the SN and an index of N^h_{intra} +1 to the CH. Then following a similar derivation, the probability that a path is formed successfully from the SN to the CH for data delivery is given by:

$$\Theta(N^h_{\text{int}\,ra}) = (\prod_{j=1}^{N^h_{\text{int}\,ra}-1}\theta_j)\times(1-Q_{rt,N_{\text{int}\,ra}(N^h_{\text{int}\,ra}+1)}) \tag{24}$$

For source redundancy, m_s SNs are used for returning sensor readings. So the failure probability[1] that all m_s SNs within a cluster fail to return sensor reading to the CH is given by:

[1] Here the failure probability is calculated based on a perfect parallel system. Section 4.1.2 treats the case in which software faults of sensors are possible and majority voting is used to calculate the failure probability.

$$Q_{fs}^{m_s} = \prod_{i=1}^{m_s} [1 - \Theta_i(N_{\text{int}ra}^h)] \tag{25}$$

Note that in each of the m_s path, distinct e_j and S_{jk} exist along each path depending on each path's traffic condition. Combining results from above, the failure probability of a source cluster not being able to return a correct response, because of either path or source failure, or both, is given by:

$$Q_f = 1 - (1 - Q_{fp}^m)(1 - Q_{fs}^{m_s}) \tag{26}$$

Therefore, for the reverse traffic the success probability for query processing is given by:

$$R_q^R = 1 - Q_f \tag{27}$$

For the forward traffic for query dissemination, we also use hop-by-hop data delivery protocol which utilizes redundancies. A query is sent from the PC to the source CH through m paths and then sent from the source CH to m_s random SNs within the cluster. Therefore, for the forward traffic, $Q^F_{rt,jk}$, θ^F_j and $\Theta^F(N_{\text{int}er}^h)$ can be similarly calculated based on Equations (19), (20), and (21) with the value of $E[S^F_{req}]$ from Equation (17). The probability of the PC failing to deliver the query to the source CH, denoted $Q_{fp}^{F,m}$, is calculated based on Equation (23) with the parameters of the forward traffic $Q^F_{rt,jk}$, θ^F_j and $\Theta^F(N_{\text{int}er}^h)$.

The probability that a path is formed successfully from the CH to a SN in its cluster for query dissemination is given by:

$$\Theta^F(N_{\text{int}ra}^h) = \left(\prod_{j=1}^{N_{\text{int}ra}^h - 1} \theta_j^F\right) \tag{28}$$

Note that for the forward traffic, we do not need the term 1-$Q_{rt,\, N^h_{inter}\,(N^h_{inter+1})}$ to account for the successful delivery of the query to the last hop since a CH can choose any m_s SNs within its cluster to be source nodes. Therefore, the failure probability that a CH fails to deliver the query

to m_s SNs within its cluster, denoted Q_{fs}^{F,m_s}, is calculated based on Equation (25) with the parameter for the forward traffic $\Theta^F(N_{\text{int}ra}^h)$.

Summarizing above, the probability that the query is failed to be delivered from the PC to the source CH and subsequently to the m_s SNs is given by:

$$Q_f^F = 1-(1-Q_{fp}^{F,m})(1-Q_{fs}^{F,m_s}) \tag{29}$$

Therefore, the success probability of the forward traffic is given by:

$$R_q^F = 1-Q_f^F \tag{30}$$

4.1.2. Software Failure

For source redundancy, m_s SNs are used for returning sensor readings. If we consider both hardware and software failures of SNs, the system will fail if the majority of SNs does not return sensor readings (due to hardware failure), or if the majority of SNs returns sensor readings incorrectly (due to software failure). Assume that all SNs have the same software failure probability, denoted by q_s, with $q_s = 1 - e^{-\lambda_{soft} t}$ as a function of time. Then to account for software failure, Equation (25) can be replaced with Equation (31) below.

$$\begin{aligned} Q_{fs}^{m_s} = & \sum_{|I|\geq\left\lceil \frac{m_s}{2}\right\rceil} \{\prod_{i\in I}[1-\Theta_i(N_{\text{int}ra}^h)]\}\{\prod_{i\notin I}\Theta_i(N_{\text{int}ra}^h)\} \\ & + \sum_{|I|\geq\left\lceil \frac{m_s}{2}\right\rceil} \{\prod_{i\notin I}[1-\Theta_i(N_{\text{int}ra}^h)]\}\{\prod_{i\in I}\Theta_i(N_{\text{int}ra}^h)\} \\ & \times\{1-[\sum_{j=\left\lceil \frac{m_s}{2}\right\rceil}^{|I|} C\left(\frac{|I|}{j}\right)(1-q_s)^j q_s^{(|I|-j)}]\} \end{aligned} \tag{31}$$

Here in Equation (31) the first expression is the probability that the majority of m_s SNs fail to return sensor readings due to hardware failure, and the second expression is the probability

that the majority of m_s SNs return sensor readings but no majority of them agrees on the same sensor reading as the output because of software failure.

4.1.3. Energy Consumption for Query Processing

In this section we calculate energy consumed for query processing for both the reverse traffic and the forward traffic. We will first derive energy expended per query for the reverse traffic E_q^R and then we will derive energy expended per query for the forward traffic E_q^F. For source redundancy, in response to a query, a SN assigned would transmit a data packet to its source CH. Since the average number of hops between a SN and its CH is given by N^h_{intra} as derived above, and a query requires the use of m_s SNs for source redundancy, the total energy required to forward data to the CH is given by:

$$E_{s-ch} = m_s N^h_{\text{int}ra}[E_T + \lambda(\pi r^2)E_R] \tag{32}$$

For path redundancy, let $E_{ch\text{-}pc}$ be the total energy consumed by the WSN to transmit sensor data from the source CH to the processing center with m paths connecting the CH to the processing center. The source CH would broadcast a copy of the data packet and all first-hop neighbors would receive. Then, among the first-hop neighbors, m nodes would broadcast again and all 2nd-hop neighbors would receive. In each of the subsequent hops on a path, only one node would broadcast and the neighbors on the next-hop would receive. Consequently, $E_{ch\text{-}pc}$ is given by:

$$\begin{aligned} E_{ch-pc} &= E_T + \lambda(\pi r^2)E_R \\ &\quad + m(N^h_{\text{int}er} - 1)[E_T + \lambda(\pi r^2)E_R] \end{aligned} \tag{33}$$

The amount of energy spent in the reverse traffic by the system, E_q^R, to answer a query that demands a source cluster to respond, using m_s SNs for source redundancy and m paths for path redundancy, is given by:

$$E_q^R = E_{ch-pc} + E_{s-ch} \tag{34}$$

For the forward traffic we calculate E_q^F by using n_q in place of n_b in Equations (1) and (2). Specifically, the amount of energy spent in the forward traffic by the system, E_q^F, to disseminate a query from the PC to the source CH and from the source CH to m_s SNs is calculated by:

$$E_q^F = N_{inter}^h[E_T + \lambda(\pi r^2)E_R] + N_{intra}^h[E_T + \lambda(\pi r^2)E_R]m_s \tag{35}$$

4.2. Energy Consumption due to Clustering

For clustering, the system would consume energy for broadcasting the announcement message and for the cluster-join process. Since p is the probability of becoming a CH, there will be pn SNs that would be broadcasting the announcement message. This announcement message will be received and retransmitted by each SN to the next hop until the TTL of the message reaches the value 0, i.e. the number of hops equals N^h_{intra}. Thus, the energy required for broadcasting is $pn[N_{intra}^h \lambda(\pi r^2)(E_T + E_R)]$. The cluster-join process will require a SN to send a message to the CH informing that it will join the cluster and the CH to send an acknowledgement to the SN. Since there are pn CHs and $(n - pn)$ SNs in the system, the energy for this is $n(E_T + E_R)$. Let the size of the message exchange be n_l. E_R and E_T will be calculated from Equations (1) and (2) with n_l in place of n_b. Let $N_{iteration}$ be the number of iteration required to execute the clustering algorithm. Then, the energy required for each execution of the clustering algorithm, $E_{clustering}$, is given by:

$$E_{clustering} = pnN_{iteration}[N_{intra}^h \lambda(\pi r^2)(E_T + E_R)] + n(E_T + E_R)] \tag{36}$$

4.3. Energy Consumption due to Status Exchange

The energy consumed due to status exchange of (E_o, e_j, Qt_{jk}) depends on if the reactive approach or the proactive approach has been adopted. We analyze both cases and their effects on MTTF. Let the size of the message exchange be n_l. The energy for reception and transmission of

status exchange message, E_R and E_T, will be calculated from Equations (1) and (2) with n_l in place of n_b.

4.3.1. Reactive Approach

Under the reactive approach, the source CH checks with m_s SNs to get the average (E_o, e_j, Qt_{jk}) when a query is processed. Also all intermediate CHs send their parameter updates to the PC. To calculate energy consumption due to status exchange, we need to know the average number of CHs between the PC and the source CH. We also need to know the average distance between two CHs.

The distance between two neighboring CHs can be calculated by:

$$d_{CH} = \frac{A}{\sqrt{N_c}} \tag{37}$$

The average number of clusters between the PC and the source CH, denoted by n_{CH}, can be calculated by:

$$n_{CH} = \frac{h}{d_{CH}}. \tag{38}$$

Thus, the distance from CH i to the PC is given by $i \times d_{CH}$, where i = [1, n_{CH}]. As a result, the number of hops between CH i to the PC, is given by:

$$N^{i}_{hop} = i\frac{d_{CH}}{r} \tag{39}$$

Therefore, the total energy consumption per query for status exchange under the reactive approach is the sum of the energy consumed for status exchange between the n_{CH} CHs and the PC, and that between the source CH and all SNs in its cluster, viz.,

$$E^{reactive}_{status} = [\sum_{i=1}^{n_{CH}} N^{i}_{hop}(E_T + E_R)] + n_s N^{h}_{intra}(E_T + E_R) \tag{40}$$

4.3.2. Proactive Approach

Under the proactive approach, every CH sends status exchange information to other CHs periodically, say, in every beacon interval denoted by T_{beacon}. The energy expended for this periodic status exchange activity per beacon interval under the proactive approach is given by:

$$E_{CH-PC}^{proactive} = C\binom{N_c}{2} N_{inter}^{h}(E_T + E_R) = \frac{N_c(N_c+1)}{2} N_{inter}^{h}(E_T + E_R) \quad (41)$$

where $C(N_c, 2)$ is the number of CH pairs in the system. All SNs in a cluster also send a status exchange message to the CH periodically. The energy expended per beacon interval for this is given by:

$$E_{SN-CH}^{proactive} = N_{intra}^{h} n_s (E_T + E_R) \quad (42)$$

Therefore, the total energy consumption for status exchange per beacon interval under the proactive approach is given by:

$$E_{status}^{proactive} = \frac{N_c(N_c+1)}{2} N_{inter}^{h}(E_T + E_R) + N_{intra}^{h} n_s (E_T + E_R) \quad (43)$$

4.4. System MTTF

Our objective is to find the best redundancy level represented by m and m_s that would satisfy the query reliability and timeliness requirements while maximizing MTTF, when given a set of system parameter values characterizing the application and network conditions. That is, if T_{req} and R_{req} are the timing and reliability requirements of a query, then we determine the best combination of (m, m_s) such that the MTTF is maximized, subject to the constraint:

$$R_q > R_{req} \quad (44)$$

Note that the constraint given above implies the timing requirement is satisfied based on how we derive R_q in Equation (3).

From a user's perspective, if the user does not experience a response returned within the specified real-time constraint, the system is considered to have failed. We define a metric called the mean time to failure (MTTF) of the sensor system that considers this failure definition. Specifically, we define the MTTF of a sensor data system as the average number of queries that the system is able to answer correctly before it fails, with the failure caused by either channel or sensor faults (such that a response is not delivered within the real-time deadline), or energy depletion.

When m paths and m_s SNs are used to achieve R_q in order to satisfy Condition (44), the amount of energy consumed is given by E_q given in Equation (4). Consider for the time being that the system fails due to energy depletion only. Then, the system fails when the system's energy falls below $E_{threshold}$. Let the potential maximum lifetime of the system be denoted by T_{life}. There are three sources of energy consumption: query processing, periodic clustering and status exchange. Consider the case in which queries arrive at the system with rate λ_q. The energy consumed due to query processing is given by $E_q \lambda_q T_{life}$ where $\lambda_q T_{life}$ is the maximum number of queries the system can possibly process during its lifetime. On the other hand, the energy expended due to the execution of the periodic clustering algorithm is given by $E_{clustering} T_{life} / T_{clustering}$ where $T_{life} / T_{clustering}$ is the number of times the clustering algorithm is executed during the system lifetime. For the reactive approach, the energy consumed due to status exchange is given by the energy consumed per query as given in Equation (40) multiplied with the number of queries experienced during the system lifetime, i.e., $\lambda_q T_{life} E_{status}^{reactive}$. Thus, T_{life} for the reactive approach can be calculated as follows:

$$\lambda_q T_{life} (E_{status}^{reactive} + E_q) + E_{clustering} \frac{T_{life}}{T_{clustering}} = E_{initial} - E_{threshold} \quad (45)$$

For the proactive approach, the energy consumed due to status exchange is given by the energy consumed per beacon interval as given by Equation (43) multiplied with the number of beacon intervals during the system lifetime, i.e., $E_{status}^{proactive} \frac{T_{life}}{T_{beacon}}$, where T_{life}/T_{beacon} is the number of beacon intervals experienced before system failure under the proactive approach. Thus, T_{life} for the proactive approach can be calculated as follows:

$$\lambda_q E_q T_{life} + E_{clustering} \frac{T_{life}}{T_{clustering}} + E_{status}^{proactive} \frac{T_{life}}{T_{beacon}} = E_{initial} - E_{threshold} \quad (46)$$

The maximum number of queries that the system is able to sustain before running out its energy, denoted by N_q, is given by:

$$N_q = \lambda_q T_{life} \quad (47)$$

Since the system is able to answer N_q queries before energy depletion, each with the reliability of R_q, the MTTF of the system is the expected number of queries that the system can answer without experiencing a failure with the upper bound of N_q, i.e.,

$$MTTF = \sum_{i=1}^{N_q - 1} i R_q^{\,i} (1 - R_q) + \quad N_q R_q^{N_q} \quad (48)$$

This MTTF metric can be translated into a more classic "system lifetime" metric with the unit of time, i.e., mean lifetime to failure (MLTF), as follows:

$$MLTF = \frac{MTTF}{\lambda_q} \quad (49)$$

Note that *MLTF* is different from T_{life} in that *MLTF* is the system lifetime, while T_{life} is the potential maximum system lifetime, i.e., it is the maximum lifetime before energy depletion if every query is processed reliably and successfully to satisfy the query QoS requirement.

4.5. Energy Consumption for Rebuilding the MMS Table

Let T_{table} be the periodical interval for rebuilding the MMS table. Let E_{table} be the energy spent each time the table is rebuilt. This energy affects the system lifetime. Specifically, T_{life} for the reactive approach now can be calculated as follows:

$$\lambda_q T_{life}(E_{status}^{reactive} + E_q) + E_{clustering} \frac{T_{life}}{T_{clustering}} + E_{table} \frac{T_{life}}{T_{table}} = E_{initial} - E_{threshold} \quad (50)$$

Here a term has been added on the left hand side to consider the energy spent to periodic rebuild the MMS table.

Similarly, T_{life} for the proactive approach can be calculated as follows:

$$\lambda_q E_q T_{life} + E_{clustering} \frac{T_{life}}{T_{clustering}} + E_{status}^{proactive} \frac{T_{life}}{T_{beacon}} + E_{table} \frac{T_{life}}{T_{table}} = E_{initial} - E_{threshold} \quad (51)$$

T_{life} calculated this way can then be used to calculate MTTF and to rebuild MMS tables based on Equation (48), listing MTTF as a function m, m_s, e_j, $Q_{t,jk}$.

4.6. Network Dynamics

The AFTQC algorithm now is extended to deal with network dynamics that cause changes to density of nodes (λ), number of sensor nodes (n), residual energy of nodes (E_o), and radio range (r) because of energy consumption and failure of nodes, and change of node connectivity.

One possible solution would be to build a larger table at design time listing optimal (m, m_s) not only as a function of (e_j, $Q_{t,jk}$) but also as a function of (λ, E_o, r). This solution depends on if the storage capability of SNs is able to hold larger tables; it will affect energy consumption as well. We dispose of this solution because of the current limited capability of SNs.

Our solution is to periodically obtain information about these dynamic model parameters, and then apply the analytical formulas derived to rebuild the best (m, m_s) table dynamically. Note that in cases where the capabilities of SNs are insufficient to perform the computation to rebuild the table (e.g., in a SUN Ultra 10 machine the computation takes about 1 minute), a base station if available can help rebuild the table and broadcast the table to CHs. The energy consumption by CHs to receive the table in this case will need to be considered. The base station in this case does not involve in query processing but will perform status exchange with CHs to rebuild the table periodically.

Here we provide an analysis for dealing with network dynamics due to node failures and change of node connectivity. Network dynamics due to node failures can be modeled by modeling the failure time of a sensor node as a random variable, i.e., exponentially distributed with a failure rate of λ_f. That is, the failure probability of a SN at time t can be calculated by:

$$q(t) = 1 - e^{-\lambda_f t} \tag{52}$$

This assumption is justified for hardware systems obeying the exponential failure law. Due to sensor node failures, the WSN needs to dynamically change radio range (r) to maintain connectivity. From [76], to maintain connectivity defined as $1 - \varepsilon$, the following relationship holds:

$$\lambda \geq \frac{1}{\pi\theta^2(1-q)r^2}\log(\frac{1}{\varepsilon\gamma^2 r^2}) \tag{53}$$

where λ is the SN density, q is the failure probability, γ and θ are parameters with $\gamma+2\theta = 1$, and ε is a very small number indicating the tolerance. Let $n(t)$ be the number of SNs at time t, given by:

$$n(t) = n \times [1 - q(t)] \tag{54}$$

Let $\lambda(t)$ be the sensor density at time t, given by:

$$\lambda(t) = \frac{n(t)}{A^2} \tag{55}$$

where A is the side of the terrain.

Plugging the expressions for $q(t)$ and $\lambda(t)$ into Inequality (53). We obtain the minimum radio range $r(t)$ for maintaining connectivity of the WSN at time t from the following equation:

$$\frac{ne^{-\lambda_f t}}{A^2} \geq \frac{1}{\pi\theta^2 e^{-\lambda_f t} r(t)^2}\log(\frac{1}{\varepsilon\gamma^2 r(t)^2}) \tag{56}$$

The residual energy per node can be calculated from the following equation:

$$E_0(t) = E_{residual}(t) / n(t) \tag{57}$$

where $E_{residual}(t)$ is the residual energy of the system at time t to be derived below.

This dynamic information regarding $\lambda(t)$, $n(t)$, $q(t)$ and $r(t)$, along with the average residual energy per SN, E_0(t), can serve as input to the periodical process for rebuilding a MMS lookup table based on Equation (48) derived earlier, listing MTTF as a function m, m_s, e_j, $Q_{t,jk}$.

Specifically, the MMS table is rebuilt periodically with $\lambda(t)$, $n(t)$, $q(t)$, $r(t)$, and $E_o(t)$ being updated periodically based on Equations (52) - (57). Note that the updates to these parameters are done periodically at discrete points with T_{table} being the period.

Similarly $E_q(t)$, $E_{clustering}(t)$, $E_{status}^{reactive}(t)$, $E_{status}^{proactive}(t)$, $E_{table}(t)$ can all be updated as a function of time at discrete points with T_{table} being the period. Thus, by rewriting Equation (50) taking into consideration of network dynamics, $E_{residual}(t)$ under reactive can be calculated by:

$$E_{residual}(t) = E_{initial} - \sum_{i=0}^{t/T_{table}-1} \lambda_q T_{table}[E_q(iT_{table}) + E_{status}^{reactive}(iT_{table})] - \sum_{i=0}^{t/Tclustering-1} E_{clustering}(iT_{clustering}) - \sum_{i=0}^{t/T_{table}-1} E_{table}(iT_{table}) \quad (58)$$

Here the notations of $E_q(t)$, $E_{clustering}(t)$, $E_{status}^{reactive}(t)$, $E_{status}^{proactive}(t)$, $E_{table}(t)$ have been used to refer to the values returned at time t. In the first summation in the right hand side of the expression, there are altogether t/T_{table} periods with index i referring to the ith period, and $E_q(i\ T_{table})$ refers to the energy consumed per query during the ith period.

Similarly by rewriting Equation (51) $E_{residual}(t)$ under proactive can be calculated by:

$$E_{residual}(t) = E_{initial} - \sum_{i=0}^{t/T_{table}-1} \lambda_q T_{table} E_q(iT_{table}) - \sum_{i=0}^{t/Tclustering-1} E_{clustering}(iT_{clustering}) - \sum_{i=0}^{t/T_{beacon}-1} E_{status}^{proactive}(iT_{beacon}) - \sum_{i=0}^{t/T_{table}-1} E_{table}(iT_{table}) \quad (59)$$

The maximum system lifetime T_{life} can be calculated by finding the solutions to Equations (58) and (59) after substituting $t = T_{life}$ and $E_{residual}(t) = E_{threshold}$ for reactive and proactive methods, respectively. For example, T_{life} under proactive can be calculated by the following equation:

$$\sum_{i=0}^{T_{life}/T_{table}-1} \lambda_q T_{table} E_q(t = iT_{table}) + \sum_{i=0}^{T_{life}/Tclustering-1} E_{clustering}(t = iT_{clustering}) + \sum_{i=0}^{T_{life}/Tbeacon-1} E_{status}^{proactive}(t = iT_{clustering}) + \sum_{i=0}^{T_{life}/T_{table}-1} E_{table}(t = iT_{table}) = E_{initial} - E_{threshold} \quad (60)$$

Finally, $R_q(t)$ is calculated dynamically as a function of t based on Equation (3). The maximum system lifetime T_{life} obtained above allows us to calculate the maximum number of queries, N_q, before energy depletion, each with the reliability of $R_q(t)$. The MTTF of the system is the expected number of queries that the system can answer without experiencing a failure with the upper bound of N_q, i.e.,

$$MTTF = \sum_{i=1}^{N_q-1} i\{\prod_{j=1}^{i}[R_q(\frac{j}{\lambda_q})]\}[1-R_q(\frac{i+1}{\lambda_q})] \quad + \quad N_q\prod_{j=1}^{N_q}[R_q(\frac{j}{\lambda_q})] \tag{61}$$

where $R_q(j/\lambda_q)$ is the reliability of a query being processed at time $t = j/\lambda_q$.

To analyze the effect of network dynamics, MTTF can be shown as a function of table rebuild period T_{table}. For each T_{table} period, the density $\lambda(t)$, radio range $r(t)$ and residual energy $E_o(t)$ can be recalculated based on Equation (55), (56) and (57) by replacing t with the actual time value, i.e., T_{table}, 2 T_{table}, 3 T_{table}, etc.. The optimal m and m_s can be obtained from the lookup table. Then for each T_{table} period, we can calculate the MTTF based on the input of E_o, r, λ, m, and m_s.

4.7. Generalization

Certain assumptions have been made in the dissertation to simplify our analysis. Below we discuss how these assumptions can be relaxed.

4.7.1. Query Involving Multiple Clusters for a Response

First, the analysis performed so far is based on the assumption that only one source cluster is needed. It could be easily extended to the case where multiple source clusters are demanded. Let $P_q(k)$ be the probability that a query requires k source clusters to respond. Let $R_q(k)$ be the query success probability for a query that requires k source clusters to respond, and $E_q(k)$ be the energy consumption of the system to answer a query that requires k source clusters. The expressions for $R_q(k)$ and $E_q(k)$ can easily be derived from those based on a single source cluster, i.e., through Equations (3) and (4), respectively, based on the application requirements (e.g., the query is considered successful if all k source clusters must return sensor readings). Then E_q would be given by the expected value of $E_q(k)$ as:

$$E_q = \sum_{k=1}^{np} E_q(k) P_q(k) \tag{62}$$

The success probability of a query, R_q, would be given as:

$$R_q = \sum_{k=1}^{np} R_q(k) P_q(k) \tag{63}$$

The same analytical expression for the MTTF as given by Equation 48 with new E_q and R_q given in Equations (62) and (63) then can be used to analyze the effect of $P_q(k)$.

4.7.2. Concurrent Query Processing with Distinct QoS Requirements

We have considered the scenario in which queries are issued one at a time by a mobile user driving a car or on an aircraft collecting sensor readings of interest, so queries are processed sequentially by the WSN. Our analysis can also be applied to scenarios, in which queries arrive at the system concurrently, say, by multiple users.

For applications where concurrent queries access different source clusters, the AFTQC algorithm can be applied without modification. The reason is that the MTTF metric is based on the number of queries the system is able to satisfy before system failure; therefore, we only need to concern about the maximum number of queries, regardless of if they are one-by-one queries or concurrent queries. The MTTF value obtained afterward can be translated into the lifetime of the system by considering the rate at which queries arrive.

For applications where concurrent queries may access a cluster simultaneously, the only change to the AFTQC algorithm is that the PC will have to calibrate (e_j, $Q_{t,jk}$) information reported from the source cluster as well as those clusters between the PC and the source cluster because these clusters might be involved in concurrent query processing. In both cases, concurrent query processing would be reflected by the increase of query arrival rate, which can be used to translate the MTTF metric into the conventional system lifetime metric. Specifically, by simply measuring e_j (transmission failure probability experienced by SN_j) and $Q_{t,jk}$ (speed violation probability if the packet is forwarded from SN_j to SN_k) to properly account for the noise and interference introduced due to simultaneous transmission of data packets by SNs. The reason is that the MTTF metric used is based on the number of queries that the system is able to service before it fails, so it does not matter whether queries are processed sequentially or

concurrently, as long as the noise and interference introduced due to simultaneous transmission of data packets by SNs have been properly accounted for in calculating the query success probability (R_q) and energy consumption (E_q).

We have used the average QoS (reliability and timeliness) requirements for all queries in the analysis. In reality, queries may be in different service classes and thus have different QoS requirements. The analysis can be extended to handle this more general case by considering the probability of a query being in a particular QoS class and computing the weighted R_q and E_q of a query, and consequently the MTTF of the system. For example a timeliness requirement can be (1sec, 5sec, 10sec) and a reliability requirement can be (0.999, 0.99, 0.9) so there will be nine QoS classes. For each QoS class, say, class i, we apply the analysis method to calculate $R_{q,i}$ and $E_{q,i}$ for class i only as in Chapter 4, Then given knowledge of the probability that a query is in class i, $PQoS_i$, we can calculate the expected reliability and energy consumption per query, $\overline{R_q}$ and $\overline{E_q}$, as:

$$\overline{R_q} = \sum_i PQoS_i \times R_{q,i} \tag{64}$$

$$\overline{E_q} = \sum_i PQoS_i \times E_{q,i} \tag{65}$$

Then the MTTF calculation can use $\overline{R_q}$ and $\overline{E_q}$ instead of R_q and E_q.

4.7.3. Data Aggregation

So far we assume that a source CH will not aggregate data. A source CH may receive up to m_s redundant packets due to source redundancy but will just forward the first one received to the PC. Thus, the data packet size is the same. For more sophisticated scenarios, conceivably the CH could also aggregate data for query processing and the size of the aggregate packet may be larger than the average data packet size.

We extend the analysis to deal with data aggregation in two ways. The first is to set a larger size for the aggregated packet that would be transmitted from a source CH to the PC. This will have the effect of favoring the use of a smaller number of redundant paths (i.e., m) because

more energy would be expended to transmit aggregate packets from the source CH to the monitoring node. The second is for the CH to collect a majority of sensor readings from its sensors before data are aggregated and transmitted to the PC. The analysis of data aggregation thus in effect is the same as that we have performed for SN software faults in Section 4.1.2 requiring a majority of sensors to return correct sensor readings.

4.8. Acknowledgement with Retransmission

The nature of wireless communication is unreliable. Therefore, to improve the reliability of packet delivery, a commonly used mechanism is to use acknowledgements to confirm the reception of a packet and to retransmit if the packet is not received. However, there is a tradeoff between reliability and energy consumption. This acknowledgement with retransmission approach increases the energy consumption and the transmission delay, and may cause the end-to-end deadline to be violated. In this section we consider a design of the AFTQC algorithm based on acknowledgement with retransmission to contrast with the design of AFTQC without retransmission. The acknowledgement considered in this approach is implicit in the sense that there is no explicit acknowledgement packet being sent by the receiver SN to the sender SN to save scarce bandwidth. Since the wireless medium is open, the sender SN can listen to the wireless medium to know if its next hop forwarding SN has forwarded the very same packet. If it does not detect that, the sender SN will attempt to retransmit a number of times before announcing failure.

For a sender node, say SN_j, it sets the maximum number of retries dynamically depending on the transmission error probability *ej* detected at runtime. We denote *Rj* as this maximum number of retries, calculated based on *ej* as follows:

$$R_j = \frac{1}{1-e_j} - 1 \qquad (66)$$

Intuitively, the first term in the right hand side expression denotes the average number of tries after which a success is to be expected. As a result, we set the maximum number of retries to be the same as this average number minus 1 (the initial try) to gain maximum reliability. With this set, the probability of a SN failing to relay sensor data as calculated in Equation (10) can be replaced by the following equation:

$$Q_{r,j} = 1 - [(1-q)(1-e_j^{R_j+1})] \tag{67}$$

To calculate the speed requirement at each hop in order to satisfy the end-to-end deadline requirement, we consider a slot-based scheme such that each transmission or retransmission is given a time "slot" such that the overall time needed for packet delivery along the PC-CH path and the CH-SN path does not exceed the deadline requirement, denoted by *Time**, where *Time** can be $Time^R$ or $Time^F$ depending on whether the traffic considered is the reverse or forward traffic. Since SN_j is expected to transmit $R_j + 1$ times before it experiences the first success, we allocate $R_j + 1$ slots of time to SN_j so that the time allocated to it is inversely proportional to its transmission error probability. The time allocated to each slot can be calculated as follows:

$$T_{slot} = \frac{Time^*}{\sum_{k_1=1}^{N_{inter}^h}(R_{k_1}+1) + \sum_{k_2=1}^{N_{intra}^h}(R_{k_2}+1)} \tag{68}$$

Here the denominator is the sum of all time slots along the PC-CH path and the CH-SN path. The time allocated to each hop is based on the number of expected transmissions at that hop and is given by:

$$T_{hop_j} = T_{slot}(R_j + 1) \tag{69}$$

That is, when a forwarding SN needs to retransmit more because of an unreliable link, we allocate more time to it for packet delivery. The speed requirement now is not a constant as calculated in Equation (17) and (18). Rather, it varies dynamically on a hop-by-hop basis, that is:

$$E[S_{req,j}] = \frac{r}{(R_j+1)\dfrac{Time^*}{\sum_{k_1=1}^{N_{inter}^h}(R_{k_1}+1) + \sum_{k_2=1}^{N_{intra}^h}(R_{k_2}+1)}} \tag{70}$$

This expression will be used in Equation (19) to calculate the probability that the imposed speed requirement at hop j is violated by SN_j while it transmits a packet to SN_k. All other equations in (20)-(30) are not changed. Specifically, Equation (67) and Equation (70) then can be

combined with Equations (27) and (30) for calculating the reliabilities of reverse traffic and forward traffic, respectively.

The total energy required to forward data from m_s SNs to the CH in this case will be the total energy required to forward data for all retries to the CH. Equation (32) can be replaced with the following equation:

$$E_s = [\sum_{k_2=1}^{N_{intra}^h}(R_{k_2}+1)]m_s[E_T + \lambda(\pi r^2)E_R] \tag{71}$$

The total energy consumed by the WSN to transmit sensor data from the source CH to the processing center will be the total energy for all retries from CH to the PC on all m paths. Equation (33) can be replaced with the following equation:

$$\begin{aligned} E_{ch} &= (R_1+1)[E_T + \lambda(\pi r^2)E_R] \\ &+ m\sum_{k_1=2}^{N_{inter}^h}(R_{k_1}+1)[E_T + \lambda(\pi r^2)E_R] \end{aligned} \tag{72}$$

For forward traffic, the total energy required to disseminate a query from the PC to the source CH and from the source CH to m_s SNs is calculated by:

$$\begin{aligned} E_q^F = &[\sum_{k_1=1}^{N_{inter}^h}(R_{k_1}+1)](E_T + \lambda(\pi r^2)E_R) \\ &+ [\sum_{k_2=1}^{N_{intra}^h}(R_{k_2}+1)](E_T + \lambda(\pi r^2)E_R)m_s \end{aligned} \tag{73}$$

4.9. Applicability and Summary

In this chapter, we developed mathematical models to analyze our proposed path and source redundancy fault tolerance mechanisms, which when properly employed, could achieve QoS requirements while maximizing the lifetime of query-based sensor networks. We derived the probability of successful data delivery within a real-time constraint (R_q), as well as the amount of energy consumed (E_q) per query. When given a set of parameter values characterizing the operating and workload conditions of the environment, we identified the optimal (m, m_s)

setting that would maximize the MTTF while satisfying the application QoS requirements. We also developed mathematical models to consider network dynamics and to relax assumptions made in the basic model to make it more generally applicable to WSNs with query processing capability.

To apply the results derived in this dissertation, a MMS table can be built at design time listing MTTF as a function of (m, m_s) covering a perceivable set of parameter values for T_{req}, R_{req}, e_j, $Q_{t,jk}$, λ, E_o, and r. When memory is an issue, MMS tables can be rebuilt periodically after λ, E_o, and r are determined dynamically. A MMS table in this case would list the best (m, m_s) covering a perceivable set of parameter values for T_{req}, R_{req}, e_j, and $Q_{t,jk}$ only. Dynamic parameter values such as e_j (transmission failure probability experienced by SN_j) and $Q_{t,jk}$ (speed violation probability if the packet is forwarded from SN_j to SN_k) can be predicted by using local measurements, or, alternatively collected either proactively or reactively by the CHs at the expense of energy consumption. Then, at runtime a simple table lookup could be performed at runtime to determine the optimal (m, m_s) that could satisfy the QoS requirements and maximize the MTTF of the WSN system.

Chapter 5
ANALYSIS AND NUMERICAL RESULTS

In this chapter we present numeric data to demonstrate the tradeoff between R_q and E_q and that there exists an optimal (m, m_s) set that would maximize the MTTF of the sensor system while satisfying Condition (44). Table 5.1 lists the parameters used along with their default parameters. These default values are consistent with the values used in [48], [51], [59], and [63]. Our WSN consists of 1000 sensor nodes distributed according to a Poisson process with density λ in a square area of 400m by 400m. Each SN has a transmission radio range of 40 m. The initial bandwidth of the wireless channel is 200Kb/s. Each SN has an initial energy of 10 Joule. The energy parameters used by the radio module are adopted from [59], [63]. The energy cost to run the transmitter/receiver radio circuitry per bit processed ($E_{TX\text{-}elec}$, $E_{TR\text{-}elec}$) is chosen to be 50nJ/bit. The energy used by the transmit amplifier to achieve an acceptable signal to noise ratio is chosen to be 10 pJ/bit/m^2.

While in reality e_j (transmission failure probability experienced by SN_j) and S_{jk} (progressive speed if the packet is forwarded to SN_k from SN_j) vary depending on network traffic, we consider $e_j = e$ and S_{jk} being uniformly distributed with parameters [a, b] to simplify the analysis. We do not consider energy consumption by clustering ($E_{clustering}$=0) to focus energy consumption by data forwarding only, as the inclusion of energy consumption by clustering would decrease the MTTF but would not change the trend exhibited. We vary other key parameters to study their effect on optimal (m, m_s) and MTTF.

Table 5-1: Parameters Used for the Analysis.

Parameter	**Default Value**
m	**[1 – 10]**
m_s	**[1 – 10]**
n	**1000**
n_s	**100**
q	**10^{-6}**
e	**[0.0001 – 0.1]**
r	**40 m**
f	**¼**
λ	**10 nodes/(40 x 40 m^2)**
λ_q	**1 query/min**
A	**400m**
n_b	**50 bytes**
$E_{TX\text{-}elec}$	**50 nJ/bit**
$E_{TR\text{-}elec}$	**50 nJ/bit**
ε_{amp}	**10 pJ/bit/m^2**
E_o	**10 Joule**
$E_{threshold}$	**0 Joule**
$N_{iteration}$	**3**
$T_{clustering}$	**[5 – 20] sec**
T_{req}	**[0.3 – 1.0] sec**
T_{beacon}	**5 sec – 5 min**
B	**200Kb/s – 1Mb/s**
θ	**1/3**
γ	**1/3**
ε	**10^{-3}**
T_{table}	**5 sec – 150 sec**
E_{table}	**0.0001 – 0.01 J**
λ_f	**0.00001 – 0.001**

5.1. MTTF Analysis

Table 5.2 summarizes the optimal (m, m_s) set that would maximize the MTTF of the sensor system under the environment characterized by the set of parameter values listed in table 1. Other parameter values may generate different (m, m_s) but the trend remains the same. We see that as e increases, the system tends to use more redundancy to satisfy Condition (44) and to maximize the MTTF of the sensor system. Conversely as the real-time deadline increases, the system tends to allocate less redundancy. Most importantly, there always exists an optimal (m, m_s) set that would maximize the MTTF of the sensor system.

Table 5-2: Optimal (m, m_s) with Varying e and T_{req}.

T_{req}	e=0.0001	0.001	0.01
0.4 sec	5,5	5,5	5,6
0.5 sec	3,3	4,4	4,4
1.0 sec	2,2	3,3	4,4
2 sec	1,1	2,1	2,3
5 sec	1,1	1,1	2,3

Figure 5-1 shows a snapshot of the MTTF of the sensor system as a function of (m, m_s) with T_{req}=1.0. Two 3-D graphs are shown in Figure 5-1a to show the effect of e. The top 3-D graph is for the case in which e=0.0001 where the optimal (m, m_s) set is (2, 2) at which the MTTF is maximized. The bottom 3-D graph is for the case in which e=0.001 for which the optimal (m, m_s) set is (3, 3). We see from these two 3-D diagrams that either not enough or excessive redundancy is detrimental to the MTTF of the sensor system.

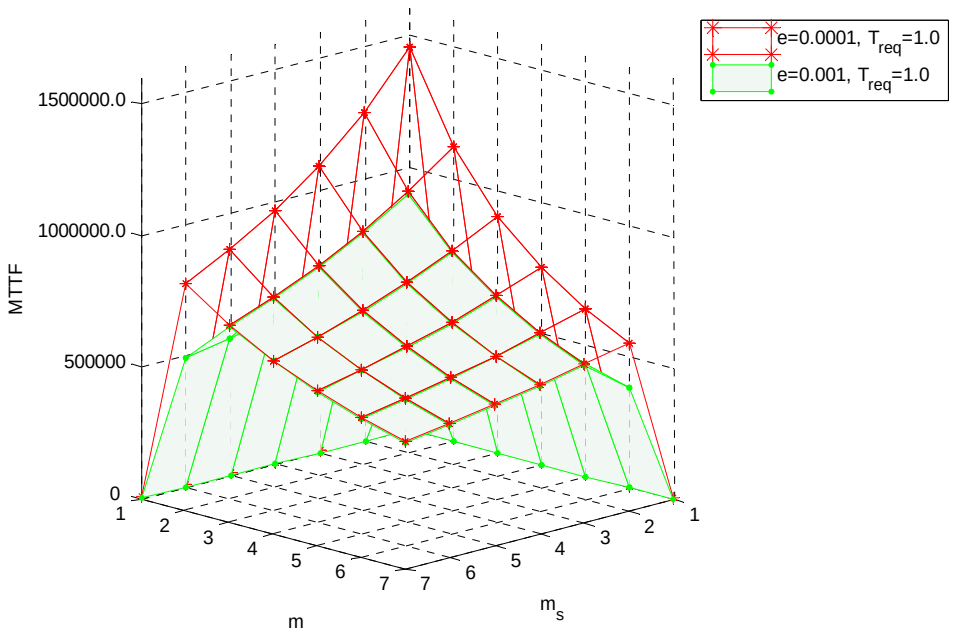

Figure 5-1: MTTF vs. (m, m_s) with T_{req}= 1 sec, e = [0.0001-0.001].

The existence of the optimal (m, m_s) set can be best understood by seeing the tradeoff between R_q and E_q as a function of (m, m_s). Figure 5-2 shows R_q vs. (m, m_s) as a function of (m, m_s). As either m or m_s increases, R_q increases. In particular, R_q is more sensitive to m because in the environment tested, the distance between the processing center and a CH (N^h_{inter}) is longer than that between the CH and a SN within a cluster (N^h_{intra}). Consequently, incorporating path redundancy (represented by m) greatly improves R_q compared with source redundancy (represented by m_s).

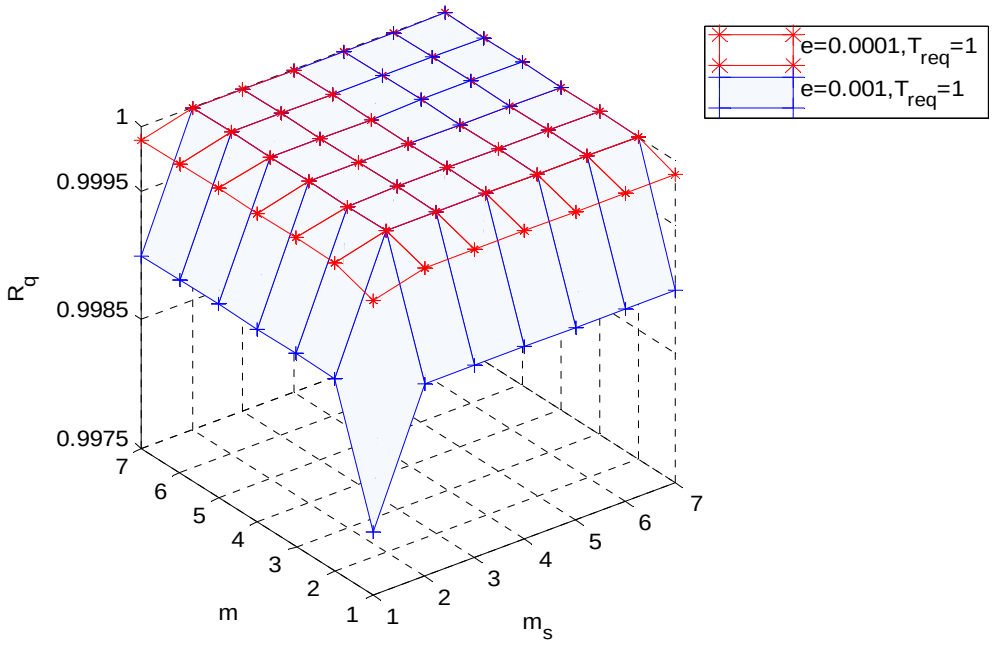

Figure 5-2: R_q vs. (m, m_s) with T_{req}=1 sec, e = [0.0001 – 0.001].

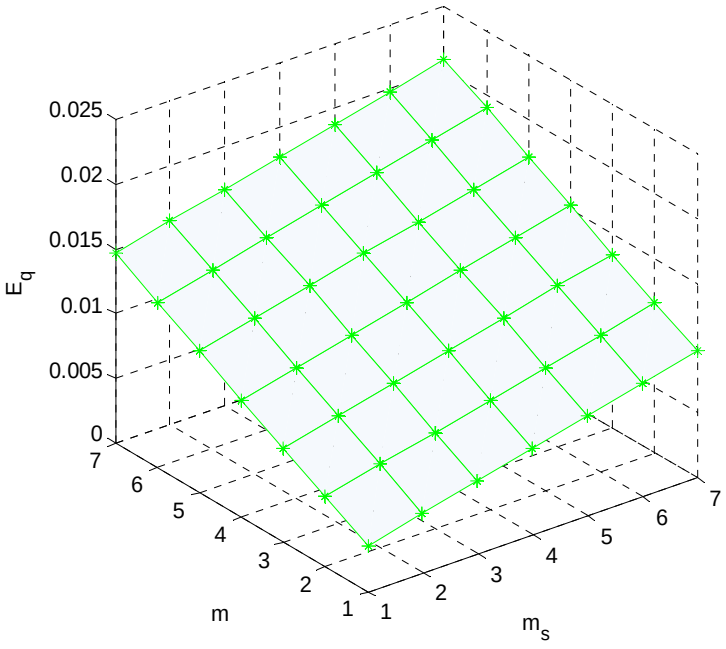

Figure 5-3: E_q vs. (m, m_s).

Correspondingly, Figure 5-3 shows the energy consumption as a function of (m, m_s). We see that the energy consumption per query is monotonically increasing as either m or m_s increases. Therefore, if more redundancy is used to answer a query, on one hand the MTTF would increase due to a higher R_q (to satisfy Condition (44)), but on the other hand the MTTF would decrease due to a high E_q. As a result, an optimal redundancy level in terms of optimal (m, m_s) exists.

Next we test the effect of the real-time deadline on MTTF. Figure 5-4 shows a snapshot of the MTTF of the sensor system as a function of (m, m_s) with e=0.0001 with varying T_{req}. The top 3-D graph is for the case in which T_{req}=1.0 for which the optimal (m, m_s) set is (2, 2) at which the MTTF is maximized. The bottom 3-D graph is for the case in which T_{req}=0.5 for which the optimal (m, m_s) set is (4, 4). In general, we observe that as T_{req} increases (less stringent real-time deadline constraints), the MTTF increases. Also the system would select less redundancy to maximize the MTTF of the system.

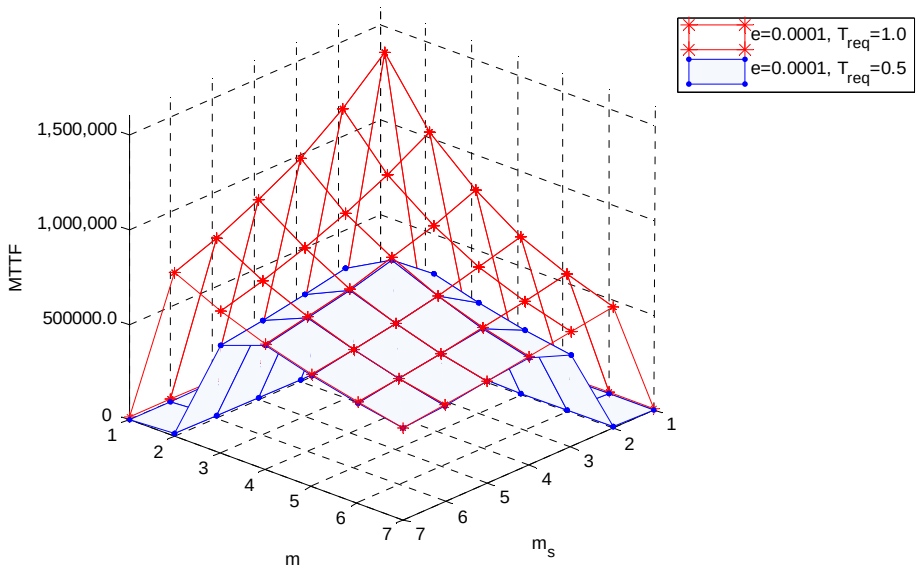

Figure 5-4: MTTF with e = 0.0001, T_{req} = [0.5 – 1.0] sec.

5.2. Tradeoff Analysis between AFTQC without Acknowledgement and with Acknowledgement

In our basic design of AFTQC, we use redundancy instead of ACK to achieve high reliability. However in extreme cases where the transmission failure reliability is low, it may be advantageous to use ACK to achieve a higher MTTF. In this section, we analyze the tradeoff between the AFTQC algorithms without acknowledgement vs. with acknowledgement and identify conditions under which one scheme may perform better than the other in terms of the system MTTF metric.

Figure 5-5 shows a snapshot of the MTTF of the sensor system as a function of (m, m_s) with T_{req}=1.0, e = 0.0001 under which AFTQC with ACK is worse than AFTQC without ACK. Both 3-D graphs show the optimal (m, m_s) set of (2, 2) at which the MTTF is maximized. The top 3-D graph is for the case of AFTQC without using acknowledgement. The bottom 3-D graph is for the case of AFTQC with acknowledgement. We see that the bottom graph has lower values of the system MTTF than the top graph. This is because when the deadline is long (T_{req} = 1 sec) and the transmission failure probability is small (e = 0.0001), the system will spend more energy in sending acknowledgements. Thus, AFTQC without ACK is better in this case.

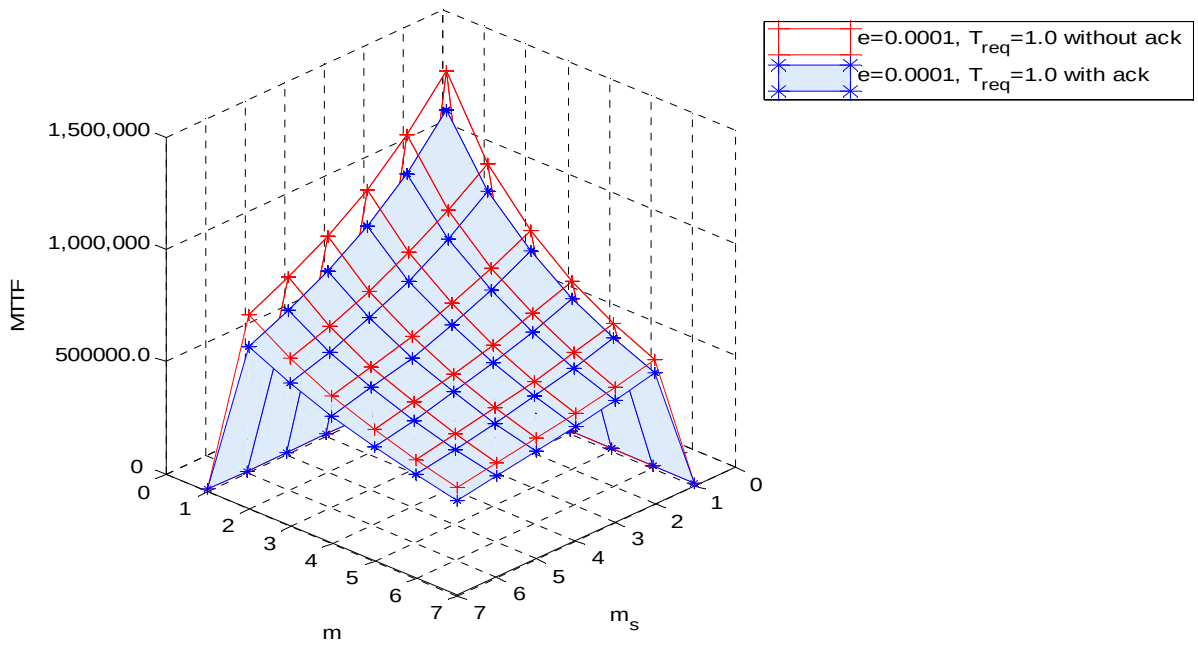

Figure 5-5: AFTQC with ACK vs. AFTQC without ACK under e = 0.0001, T_{req} = 1.0 sec.

Conversely, Figure 5.6 shows a snapshot of the MTTF of the sensor system as a function of (m, m_s) with T_{req}=0.5, e = 0.1 under which AFTQC with acknowledgement is better than AFTQC without acknowledgement. Both 3-D graphs show the optimal (m, m_s) set of (7, 9) at which the MTTF is maximized. The top 3-D graph is for the case of AFTQC with acknowledgement. The bottom 3-D graph is for the case of AFTQC without acknowledgement. We see that the top graph has much higher values of the system MTTF than the bottom graph. This is because when the deadline is short (T_{req} = 0.5 sec) and the transmission failure probability is high (e = 0.1), the gain in system reliability due to using acknowledgement outweighs the energy spending in sending acknowledgement packets.

In summary, we conclude that whether to use ACK or not depends on the transmission failure probability at the time of query processing and the response time deadline imposed. Our design allows the system to adapt to network and query conditions to dynamically determine the best scheme to use to maximize the system MTTF. Under normal conditions where the channel transmission reliability is relatively high, we expect AFTQC without acknowledgement to be our design choice.

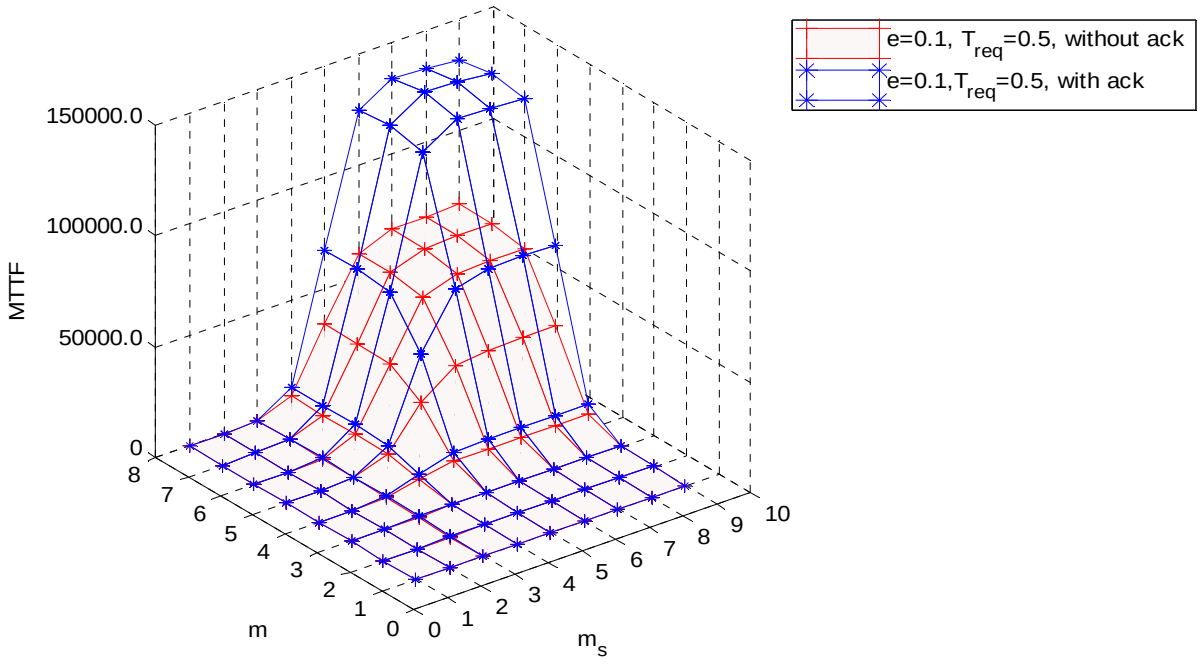

Figure 5-6: AFTQC with ACK vs. AFTQC without ACK under e = 0.1, T_{req} = 0.5 sec.

5.3. Comparison of AFTQC with Baseline

We compare our design with a baseline design in which there is no redundancy and the classic "acknowledgement and retransmission on timeout" mechanism is used. Figures 5-7 and 5-8 show a snapshot of the MTTF of the sensor system as a function of (m, m_s) in logarithmic scale in order to show the baseline case. Figure 5-7 is for the case in which e=0.0001, T_{req}=1.0. Since in this case the channel transmission reliability is relatively high, we expect AFTQC without acknowledgement performs better than AFTQC with acknowledgement. The top 3-D graph shows the MTTF under AFTQC without ACK. The middle 3-D shows that under AFTQC with ACK. The optimal (m, m_s) is (2, 2) in both graphs. The bottom 3-D graph shows the MTTF using the baseline design (labeled as m = 1, m_s = 1, baseline with ACK). We observe that AFTQC either with ACK or without ACK greatly increases the system MTTF compared with the baseline design under this set of parameter values characterizing the WSN. The effect is especially pronounced when AFTQC operates at the optimal (m, m_s) identified. Moreover, even when the WSN is extremely reliable, i.e., when e and q are extremely small, so that the optimal (m, m_s) at (1, 1), AFTQC without ACK still yields a higher MTTF than the baseline system because of energy saving in not using acknowledgements.

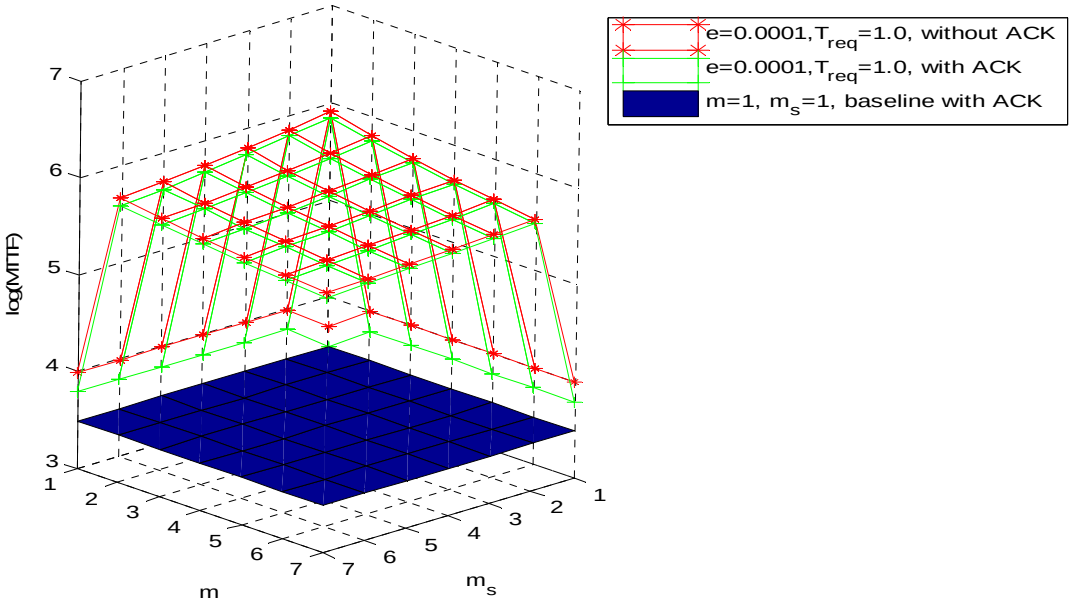

Figure 5-7: AFTQC vs. Baseline with T_{req}= 1 sec, e = 0.0001 in Logarithmic Scale.

Next we consider a case in which the channel transmission reliability is relatively low so that AFTQC with ACK is expected to perform better than AFTQC without ACK. Figure 5-8 is for the case in which e=0.1, T_{req}=1.0. The top 3-D graph shows the MTTF under AFTQC with ACK. The optimal (m, m_s) is (6, 7) in this case. The middle 3-D shows that under AFTQC without ACK. The optimal (m, m_s) is (7, 8) in this case. As expected, the system requires more path and source redundancies under AFTQC without ACK when the network is not reliable. The bottom 3-D graph shows the MTTF using the baseline design (labeled as m = 1, m_s = 1, baseline with ACK). We observe that when the network is not very reliable, i.e. when e is large, as expected AFTQC with ACK performs better than AFTQC without ACK. The baseline scheme performs marginally better than AFTQC without ACK only when (m = 1, m_s = 1). At other (m, m_s) settings, both AFTQC with ACK and AFTQC without ACK significantly outperform the baseline scheme, especially at optimal (m, m_s) values identified. We again observe that using redundancies greatly improves the system MTTF.

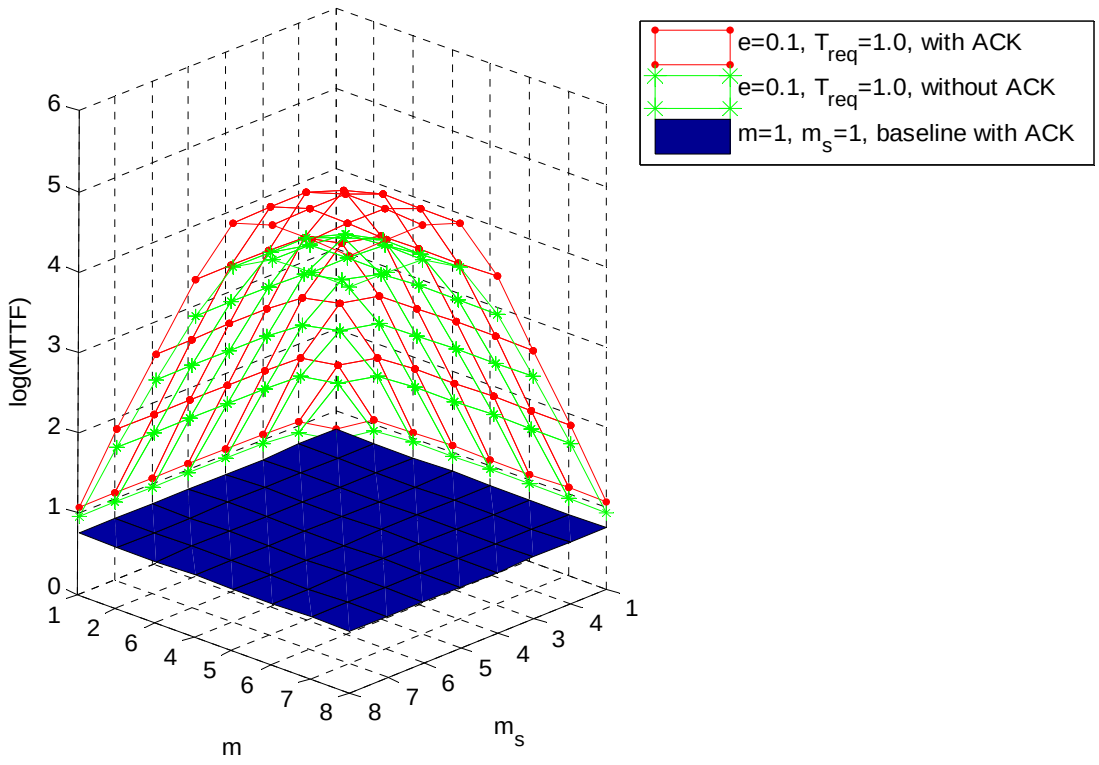

Figure 5-8: AFTQC vs. Baseline with T_{req}= 1 sec, e = 0.1 in Logarithmic Scale.

5.4. Effect of Clustering on MTTF

In this section we analyze the effect of clustering on the proposed algorithm. We also analyze the effect of different clustering intervals on the system MTTF.

Figure 5.9 shows a snapshot of the MTTF of the WSN system as a function of (m, m_s) with T_{req}=1.0, e = 0.0001 to show the effect of clustering. All 3-D graphs show the optimal (m, m_s) set of (2, 2) at which the MTTF is maximized. The top 3-D graph shows the baseline case in which the energy used for clustering is zero, i.e., $E_{clustering}$=0. The second 3-D graph is for the case when the clustering interval $T_{clustering}$ = 20 sec. The energy consumed $E_{clustering}$ is then calculated based on Equation (36). The third 3-D graph is the case when the clustering interval $T_{clustering}$ = 5 sec.

We see that when the clustering interval is short ($T_{clustering}$ = 5 sec), the MTTF values are lower than the baseline case. This is because the energy consumption by the clustering algorithm is significant in this case. When the clustering interval is sufficiently long ($T_{clustering}$ = 20 sec), the system achieves about the same values of system MTTF as the baseline case. In this case, the energy consumption by the clustering algorithm is small and does not significantly affect the system MTTF.

Finally we note that the MTTF curves for all three cases show the same trend with respect to (m, m_s) with the optimal set at (2, 2) and that the optimal (m, m_s) set is relatively insensitive to the energy used by the clustering algorithm. This is due to the assumption that clustering is executed frequent enough to maintain perfect rotation of CHs, so the frequency of clustering will only affect the total energy consumed but will not affect the optimal (m, m_s) set selected. In Chapter 6, we conduct a simulation study to identify the frequency of clustering under which the assumption is justified, and compare simulation vs. analytical results.

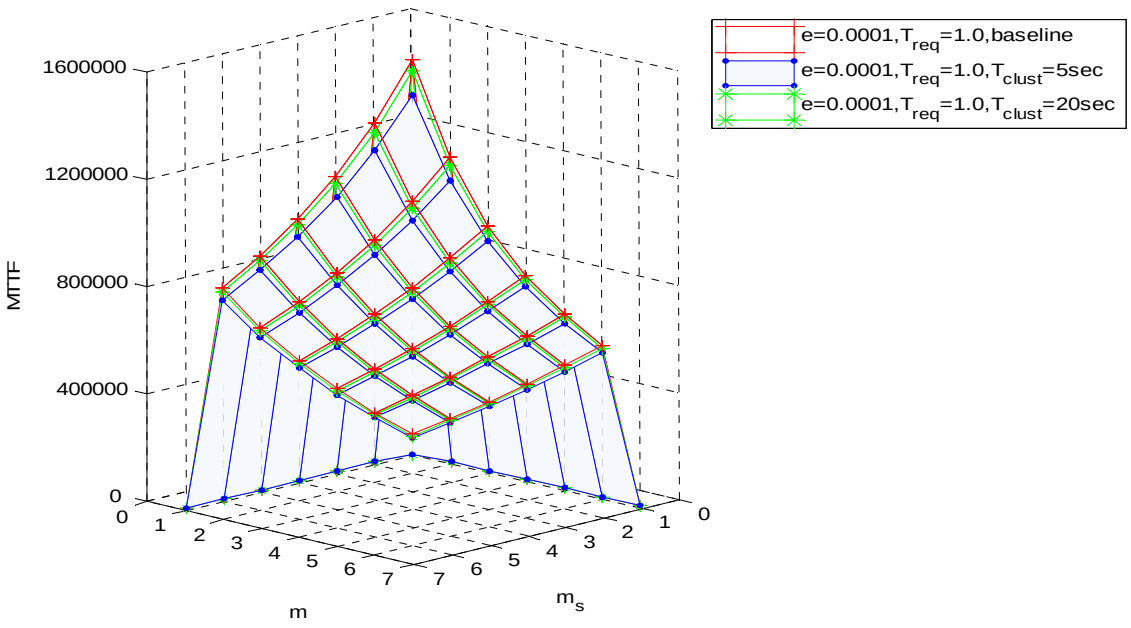

Figure 5-9: Effect of Clustering Intervals on MTTF with e = 0.0001, T_{req} = 1.0 sec.

5.5. AFTQC with Forward Traffic

So far we have presented numerical results considering the reverse traffic only. Certainly for query-based WSNs, both query dissemination (in the forward traffic) and data forwarding (in the reverse traffic) must be executed successfully by the deadline for the system to be considered functioning. In this section, we present numerical results obtained as a result of applying Equations (17) and (30) for the query success probability, as well as Equations (34) and (35) for the amount of energy spent, for the reverse and forward traffic, respectively. We demonstrate that when both forward and reverse traffics are considered, there also exists an optimal (m, m_s) set that would maximize the MTTF of the system while satisfying Condition (44).

Figure 5-10 shows a snapshot of the MTTF of the sensor system as a function of (m, m_s) with T_{req}=1.0. Two 3-D graphs are shown in Figure 5-8a. The top 3-D graph is for the case when we only include the reverse traffic in the analysis. For this case, the optimal (m, m_s) set is (2, 2) at which the MTTF is maximized. The bottom 3-D graph is for the case when we include both forward and reverse traffics in the analysis. For this case, the optimal (m, m_s) set is also (2, 2). We see from these two 3-D diagrams that including both forward and reversed traffics in the

analysis only affect the values of the system MTTF. This is because when the deadline is long, query dissemination is not failing as often, therefore the optimal and MTTF values are only affected by the energy usage for query dissemination.

Figure 5-11 shows a snapshot of the MTTF of the sensor system as a function of (m, m_s) with T_{req}=0.5. The top 3-D graph is for the case when we only include the reverse traffic in the analysis. For this case, the optimal (m, m_s) set is (3, 3) at which the MTTF is maximized. The bottom 3-D graph is for the case when we include both forward and reverse traffics in the analysis. For this case, the optimal is (5, 5). We see that when the deadline is short, query dissemination will fail more often due to the shorter time requirement for the forward traffic. Therefore the system needs to use more redundancy.

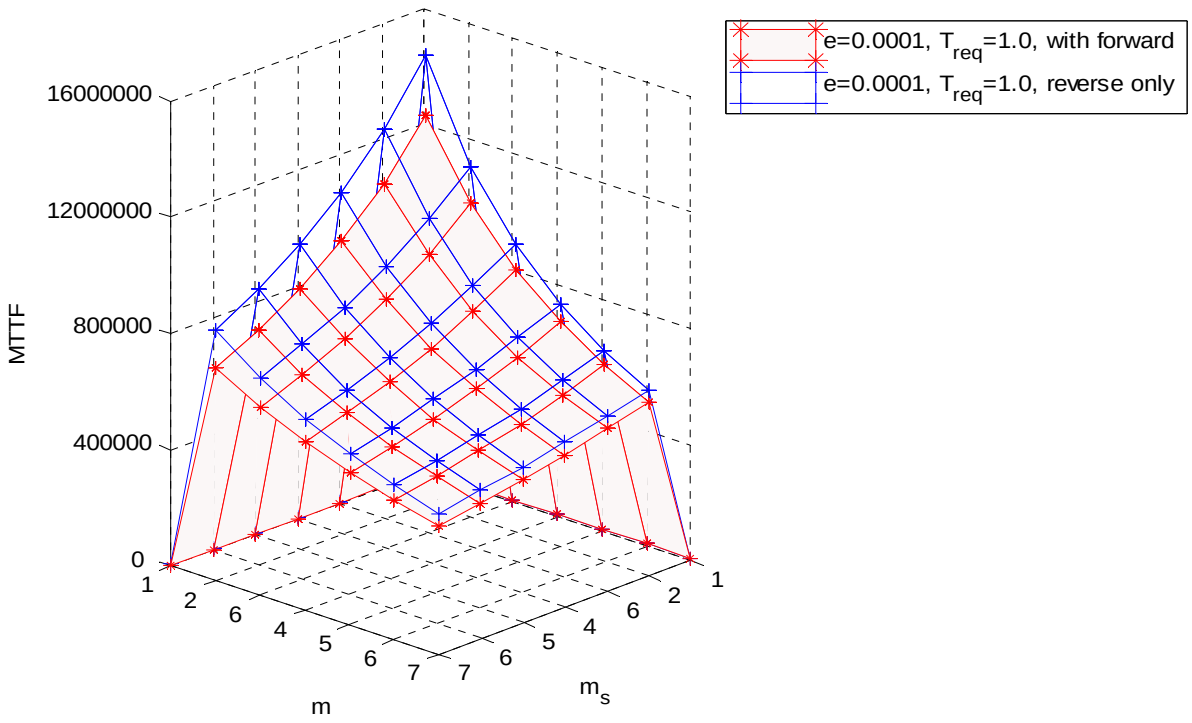

Figure 5-10: MTTF with/without Forward Traffic with e = 0.0001, T_{req} = 1.0 sec.

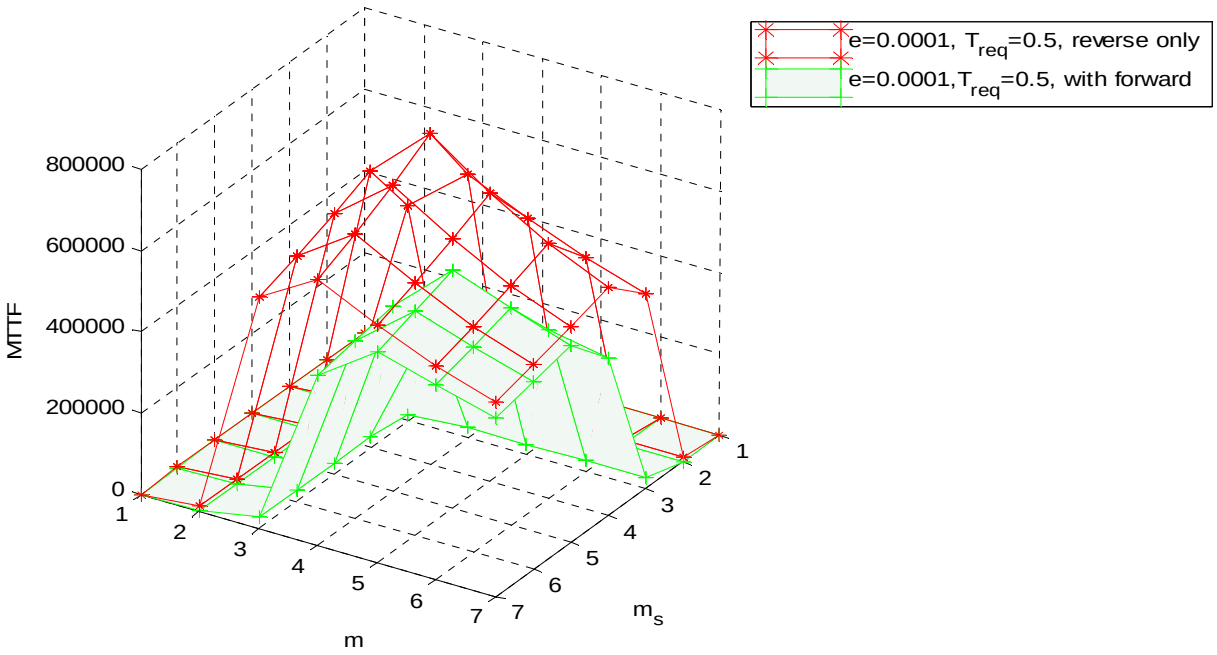

Figure 5-11: AFTQC with/without Forward Traffic with e = 0.0001, T_{req} = 0.5 sec.

5.6. AFTQC with Software Failure

In this section we analyze the effect of software faults on MTTF. Figure 5-12 shows a snapshot of the MTTF of the sensor system as a function of (m, m_s) with T_{req}=1.0 after applying Equation (31) derived in Section 4.1.2 for modeling software failure in the calculation. Figures 5-12 and 5-13 show the shift of the optimal (m, m_s) when software failure is included compare with the case when there is no software failure. Figure 5-12 is for the case in which e = 0.0001, T_{req} = 1.0. The top 3-D graph is for the case when we do not include software failure in the analysis. For this case, the optimal (m, m_s) set is (2, 2) at which the MTTF is maximized. The bottom 3-D graph is for the case when we include software failure in the analysis. For this case, the optimal (m, m_s) set is (2, 3). We see that when software failure is included in the analysis, the optimal (m, m_s) is changed from (2, 2) to (2, 3). This reflects the fact that when software faults are possible, the system tends to choose a larger number of sensor nodes to increase the probability that the majority agrees on the same sensor reading, e.g., in this case optimal m_s is changed from 2 to 3. Figure 5-13 is for the case in which e = 0.001, T_{req} = 1.0. In this case, the

optimal is changed from (3, 3) to (3, 4). Again, we see that the system chose a larger number of sensor nodes to cope with software failure.

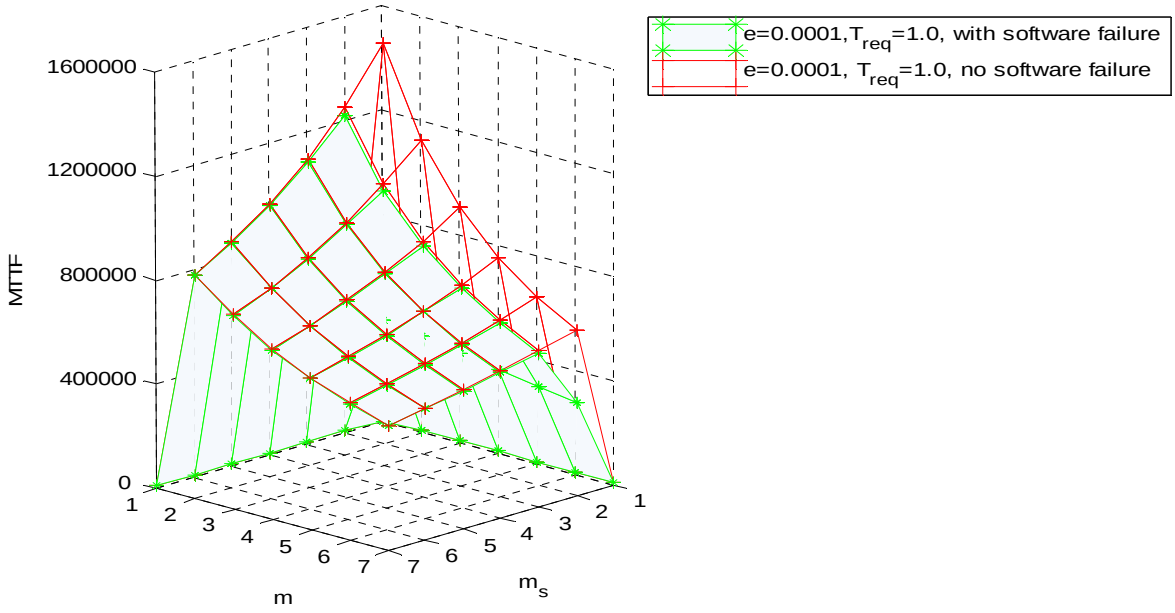

Figure 5-12: AFTQC with/without Software Failure with e = 0.0001, T_{req} = 1.0 sec.

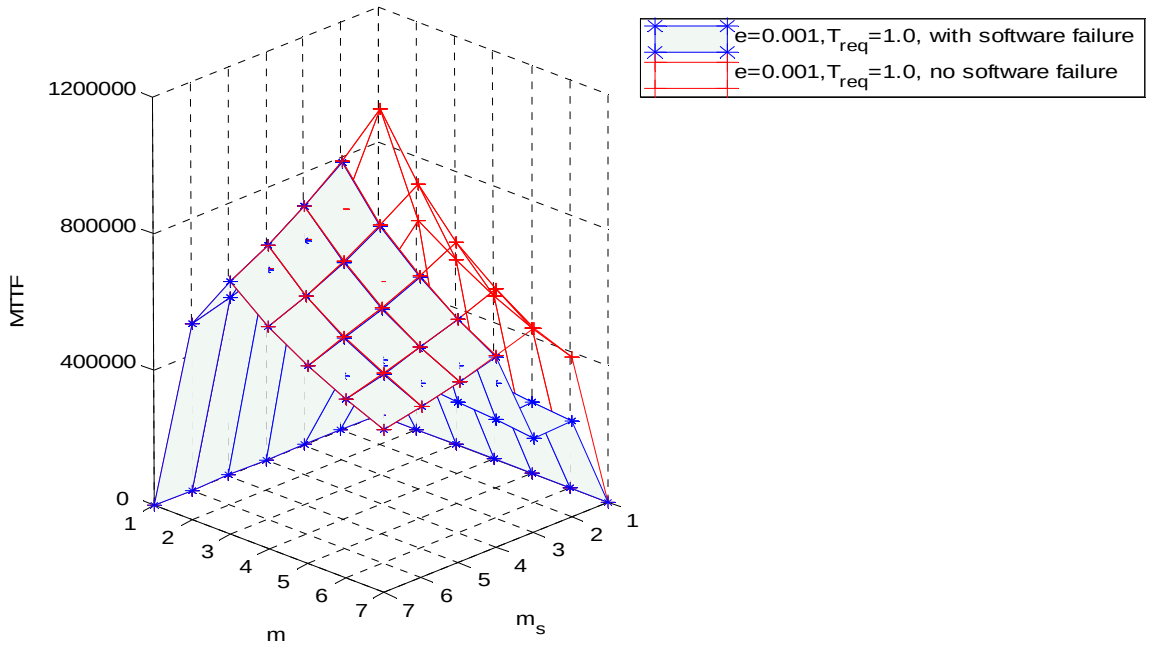

Figure 5-13: AFTQC with/without Software Failure with e = 0.001, T_{req} = 1.0 sec.

5.7. AFTQC with Multiple QoS Classes

Figure 5-14 shows a snapshot of the MTTF of the sensor system as a function of (m, m_s) for the case in which queries are in three different QoS classes with distinct deadlines. The first class has T_{req}=0.5 sec, the second has T_{req}=1 sec. and the third has T_{req}=1.5 sec. We consider that 50% of the queries arriving at the WSN are in the first class, 30% in the second class, and 20% in the third class. For this case, the optimal (m, m_s) set is (4, 2) at which the MTTF is maximized. Correspondingly, Figure 5-15 shows a 3-D graph for the case when the three QoS classes are: 1) T_{req}=1 sec, 2) T_{req}=1.5 sec and 3) T_{req}=2 sec. With the same query classification, i.e., 50% of the queries are in the first class, 30% of queries are in the second class, and 20% of queries are in the third class, we see that the optimal (m, m_s) set is (2, 1) at which the MTTF is maximized. Our observation is that when multiple QoS classes exist, the optimal (m, m_s) will be changed on a case by case basis. Also we have observed that when there are more queries with strict QoS requirements, i.e., shorter deadlines as in the first case, the system needs more redundancies than when there are fewer queries with strict QoS requirements as in the second case.

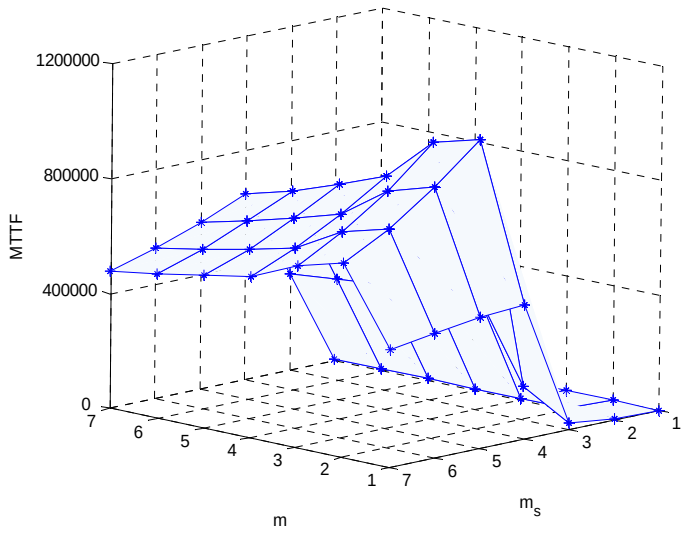

Figure 5-14: AFTQC with Multiple QoS Classes: e=0.0001, T^1_{req} = 0.5 sec, T^2_{req} = 1.0 sec, T^3_{req} = 1.5 sec.

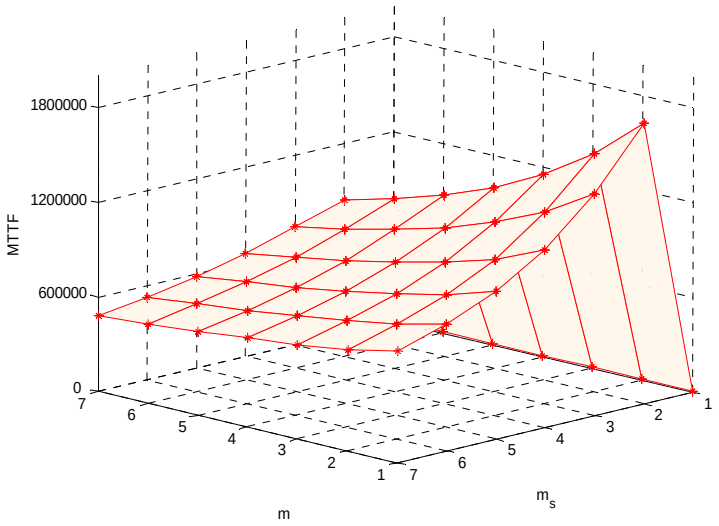

Figure 5-15: AFTQC with Multiple QoS Classes: e=0.0001, T^1_{req} = 1.0 sec, T^2_{req} = 1.5 sec, T^3_{req} = 2 sec.

5.8. Comparing Proactive AFTQC vs. Reactive AFTQC

In this section, we compare proactive AFTQC vs. reactive AFTQC to reveal design tradeoffs of these two approaches, as well as to identify conditions under which proactive AFTQC performs better than reactive AFTQC, or vice versa.

We first analyze the effect of beaconing intervals on proactive AFTQC. Figure 5.16 shows a snapshot of the MTTF of the WSN system as a function of (m, m_s) with varying beaconing frequencies T_{beacon}. All 3-D graphs show the optimal (m, m_s) set of (2, 2) at which the MTTF is maximized for the condition when e=0.0001, T_{req} = 1 sec. The top 3-D graph is for the case of using the proactive approach with a long beaconing interval (T_{beacon} = 10 min). The bottom 3-D graph is for the case with a short beaconing interval (T_{beacon} = 5 sec). As expected, when the beaconing interval is long, the proactive approach results in better MTTF. This is because the system spends less energy in sending periodic updates. The middle 3-D graph is for the case when proactive AFTQC yields about the same MTTF values as reactive AFTQC. In this case, we observe a beaconing interval of 5 min (T_{beacon} = 5 min). Next, we will set the beaconing interval to 5 min and vary other parameters to compare the proactive AFTQC vs. reactive AFTQC.

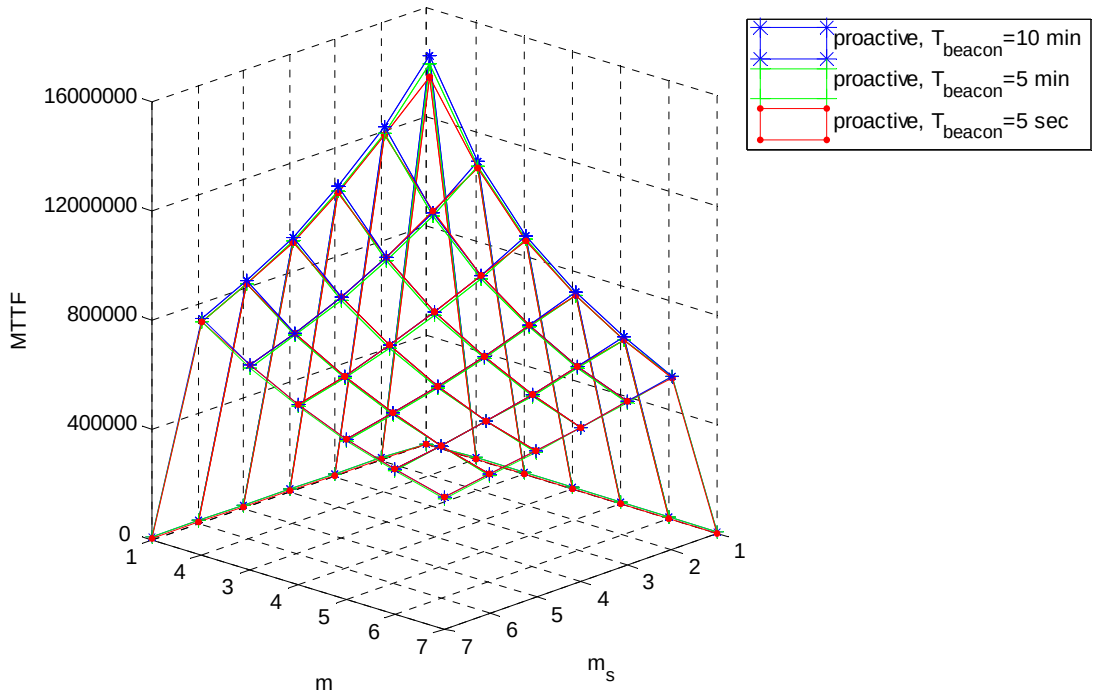

Figure 5-16: Effect of Beaconing Interval of Proactive AFTQC with $e = 0.0001$, $T_{req} = 1$ sec.

Figures 5-17 and 5-18 compare proactive AFTQC vs. reactive AFTQC head-to-head in terms of the resulting system MTTF vs. (m, m_s) in an environment in which $e = 0.01$, $T_{req} = 0.5$ sec, and $T_{beacon} = 5$ min. Figure 5-17 is for the case in which the query arrival rate is low ($\lambda_q = 1$ query/sec). Figure 5-18 is for the case in which the query arrival rate is high ($\lambda_q = 5$ queries/sec). We see that when the query arrival rate is low, the system favors the reactive approach. Conversely, when the query arrival rate is high, the system favors the proactive approach. The reason is that when the query arrival rate is high, reactive AFTQC tends to spend too much energy and time to collect node status reactively, thus lowering the system MTTF due to energy depletion or deadline violation. On the other hand, when the query arrival rate is low, proactive AFTQC tends to spend too much energy in periodic status exchange compared with reactive AFTQC, thus resulting in a lower MTTF. We also observe that the optimal (m, m_s) set is rather insensitive to the proactive vs. reactive status reporting mechanism used.

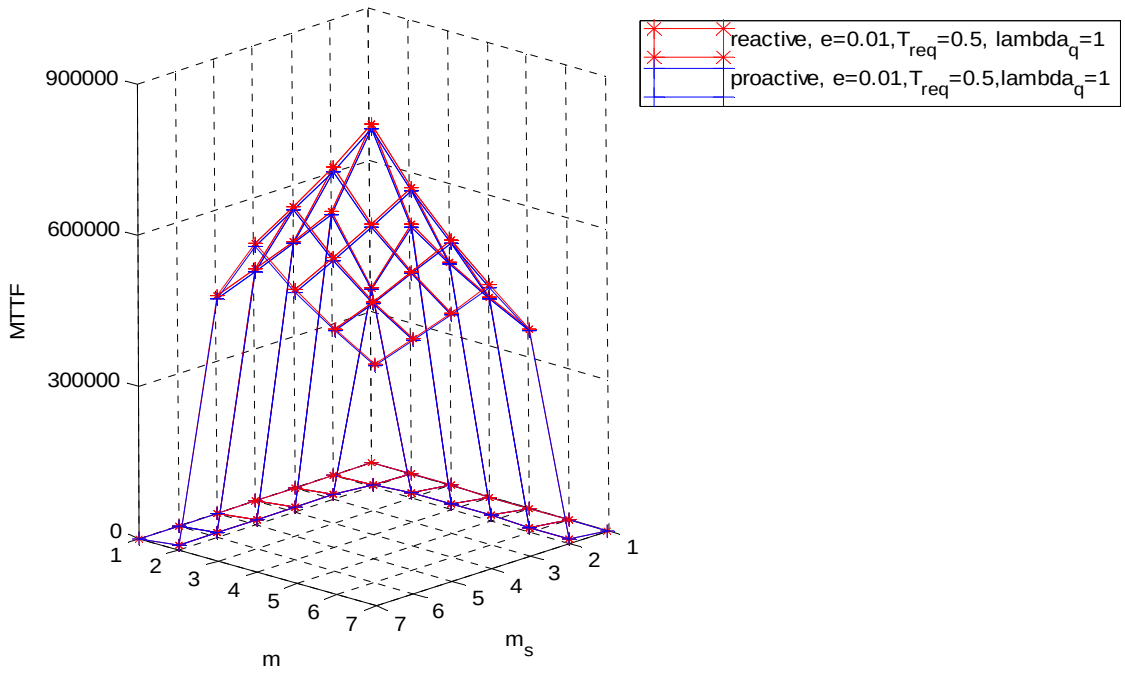

Figure 5-17: Reactive vs. Proactive AFTQC with $e = 0.01$, $T_{req} = 0.5$ sec, $\lambda_q = 1$ query/sec.

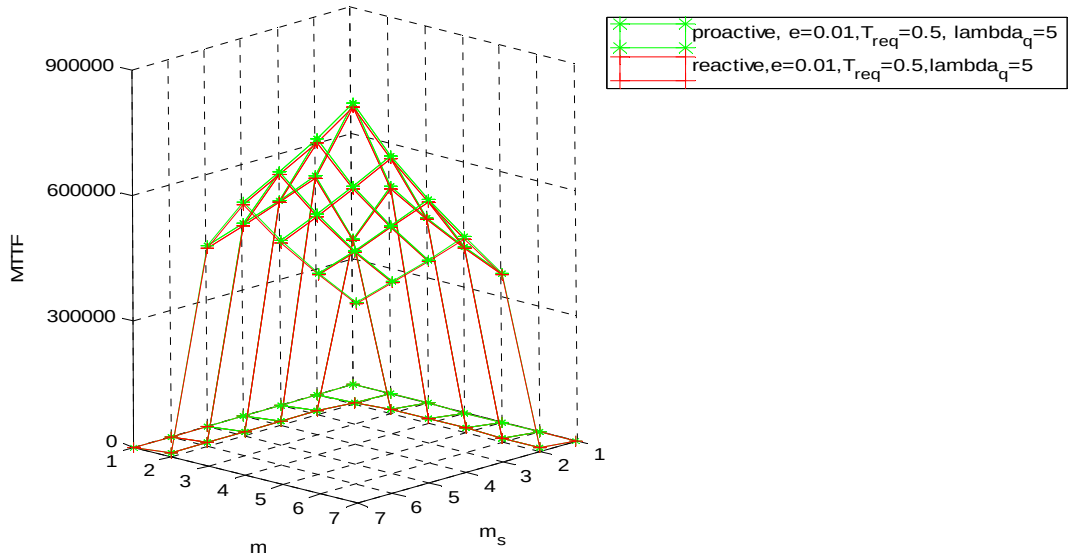

Figure 5-18: Proactive vs. Reactive AFTQC with $e = 0.01$, $T_{req} = 0.5$ sec, $\lambda_q = 5$ queries/sec.

Figures 5-19 and 5-20 compare proactive AFTQC vs. reactive AFTQC with varying deadlines in the environment in which e=0.0001, $T_{beacon} = 5$ min, $\lambda_q = 1$ query/sec. Figure 5-19 is

for the case in which T_{req} = 1 sec. In this case, we observe that the reactive AFTQC yields better MTTF values. Figure 5-20 is for the case in which T_{req} = 0.5 sec. In this case, we observe that the proactive yields better MTTF values. We see that when the deadline is short, the time taken to perform status exchange under reactive AFTQC adversely affects the chance the system is able to meet the deadline; therefore, it is better to use proactive AFTQC. In contrast, when the deadline is relatively long, the time taken to perform status exchange under the reactive approach does not significantly affect the overall deadline violation probability; therefore, it is better to use reactive AFTQC since the system spends less energy in this case.

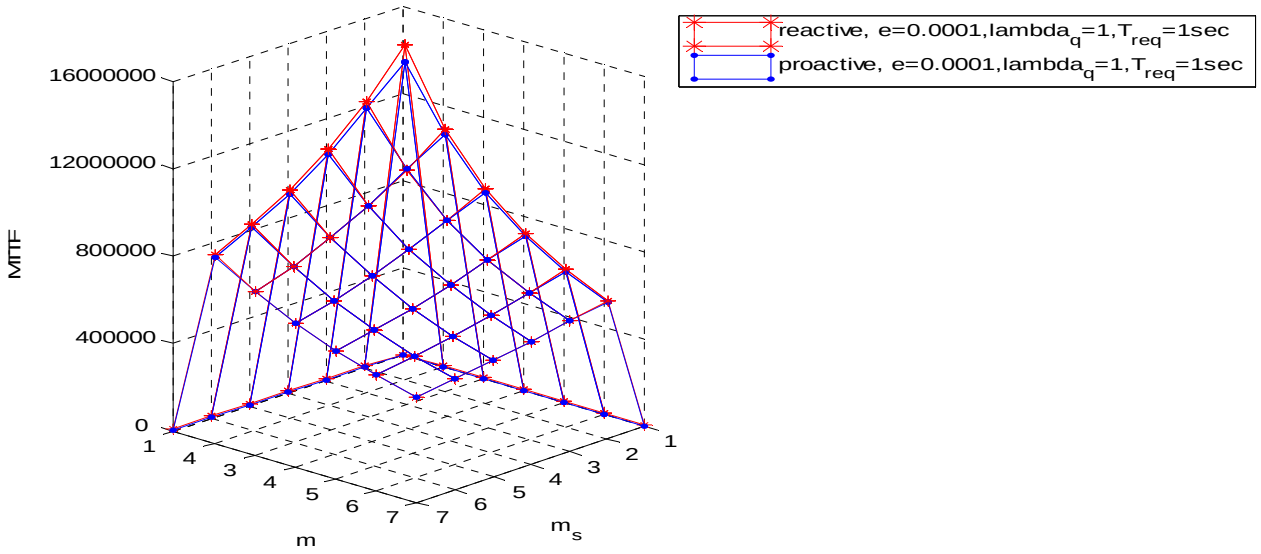

Figure 5-19: Proactive vs. Reactive AFTQC with e = 0.0001, T_{req} = 1.0 sec, λ_q = 1 queries/sec.

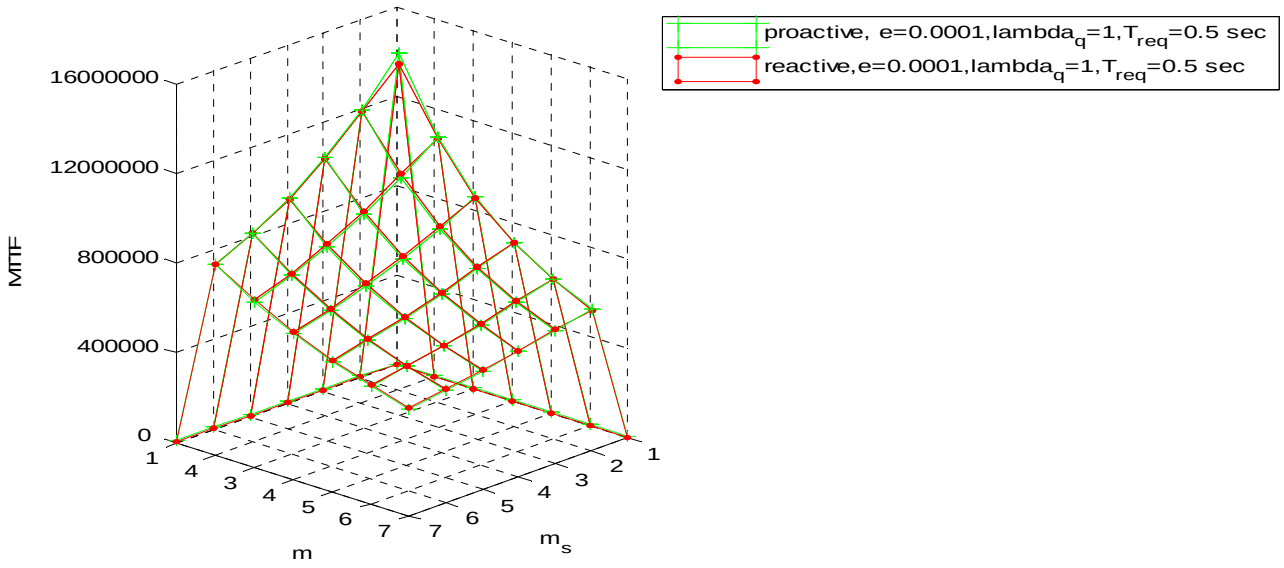

Figure 5-20: Proactive vs. Reactive AFTQC with e = 0.0001, T_{req} = 0.5 sec, λ_q = 1 queries/sec.

5.9. Effect of Bandwidth

In this section we show the effect of the wireless network bandwidth B on the optimal (m, m_s) set. The bandwidth essentially affects the MTTF though the link progressive speed. We reflect the effect of bandwidth through the (a, b) parameter. We chose a to be the lower bound of progressive speed toward the destination, i.e., a = 0, and we choose b to be the upper bound of progressive speed toward the destination, i.e., $b = r/(n_b/B)$ where n_b/B accounts for the transmission delay. A higher bandwidth results in a higher upper bound of the progressive speed, and vice versa. Two 3-D graphs are shown in Figure 5-21. The top 3-D graph is for the case when the link is fast (B = 1Mbps). For this case, the optimal (m, m_s) set obtained is (2, 2) at which the MTTF is maximized. The bottom 3-D graph is for the case when the link is slow (B = 0.2 Mbps). For this case, the optimal (m, m_s) set obtained is (3, 3). We see that when the wireless network bandwidth is changed, the optimum (m, m_s) is changed. Furthermore, as the bandwidth increases, the system tends to use less redundancy, resulting in (m, m_s) being changed from (3, 3) to (2, 2). The reason is that when the bandwidth is high, there are more paths that can satisfy the minimum speed requirement. Consequently, the system can use less redundancy to satisfy query QoS requirements.

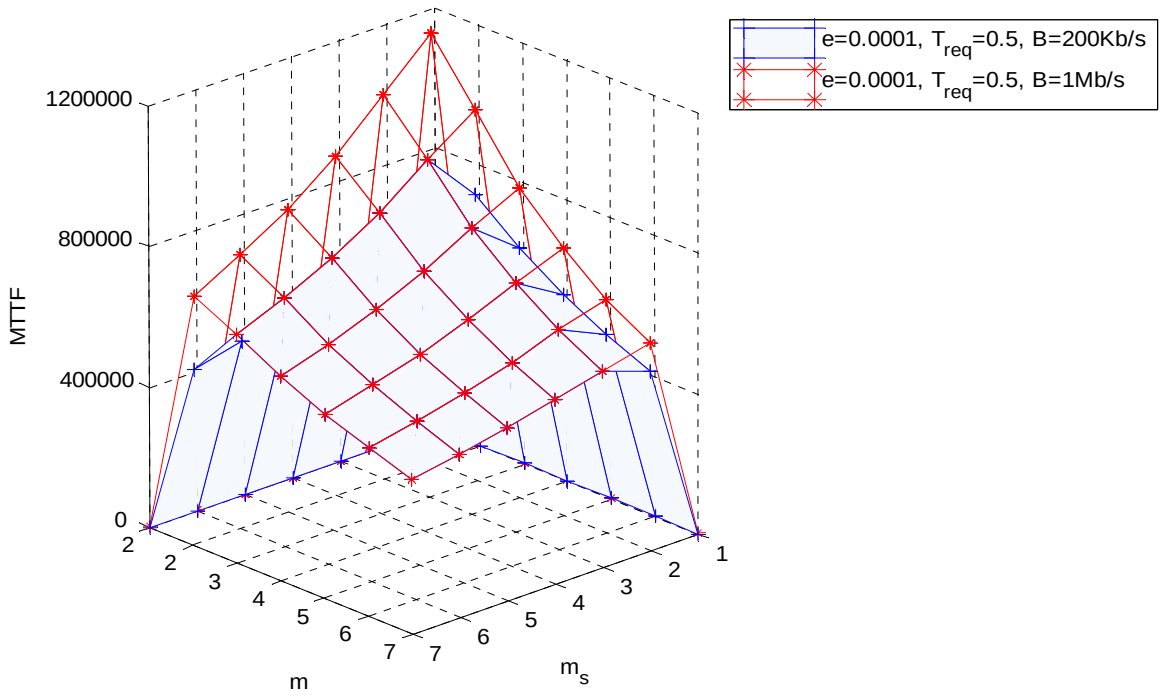

Figure 5-21: Effect of Bandwidth on Optimal (m, m_s), with e = 0.0001, T_{req} = 0.5 sec.

5.10. Effect of Network Dynamics

In this section we show the effect of network dynamics on MTTF and the optimal (m, m_s) obtained. First, we dynamically calculate the parameters $q(t)$, $n(t)$, and $\lambda(t)$ based on Equations (52), (54), and (55), respectively, and then we calculate $r(t)$ by solving Equation (56) for different T_{table} given. Solving Equation (60), we obtain the potential maximum system lifetime T_{life}. This value of T_{life} is used to calculate the expected number of queries that the system can service before failure, N_q, from Equation (47). Using N_q, we then calculate the expected MTTF based on Equation (61). The optimal (m, m_s) is obtained at discrete time points in multiple of T_{table} intervals with $q(t)$, $n(t)$, $\lambda(t)$, $r(t)$, $E_0(t)$ given as inputs at each time point. We use n = 600 and $E_{initial}$ = 0.05 J which are the same parameter values used in the simulation so that we can apply these pre-generated tables of optimal (m, m_s) at T_{table} intervals in the simulation to determine the best (m, m_s) to use in each interval as time progresses.

Figure 5-22 shows how the optimal (m, m_s) changes as time progresses in response to network dynamics for a long T_{table} interval (T_{table} = 50 sec). Figure 5-23 shows how the optimal

(m, m_s) changes as time progresses for a short T_{table} interval (T_{table} = 5 sec). We observe that when λ_f increases, m and m_s also increase as the system needs to use more redundancy to cope with sensor failures. Also when T_{table} is short, we observe more changes in the optimal (m, m_s) compare with when T_{table} is long. A short T_{table} gives us more accurate optimal (m, m_s) in trade-off of energy for table rebuilding.

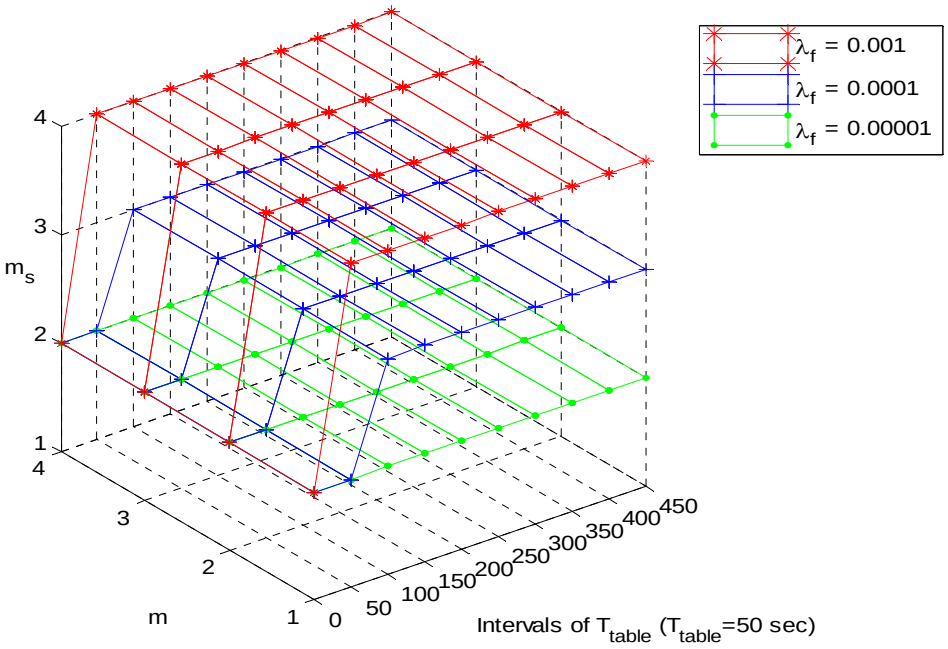

Figure 5-22: Optimal (m, m_s) vs. Time in Increment of T_{table}, with T_{table} = 50 sec, λ_f=[0.00001-0.001].

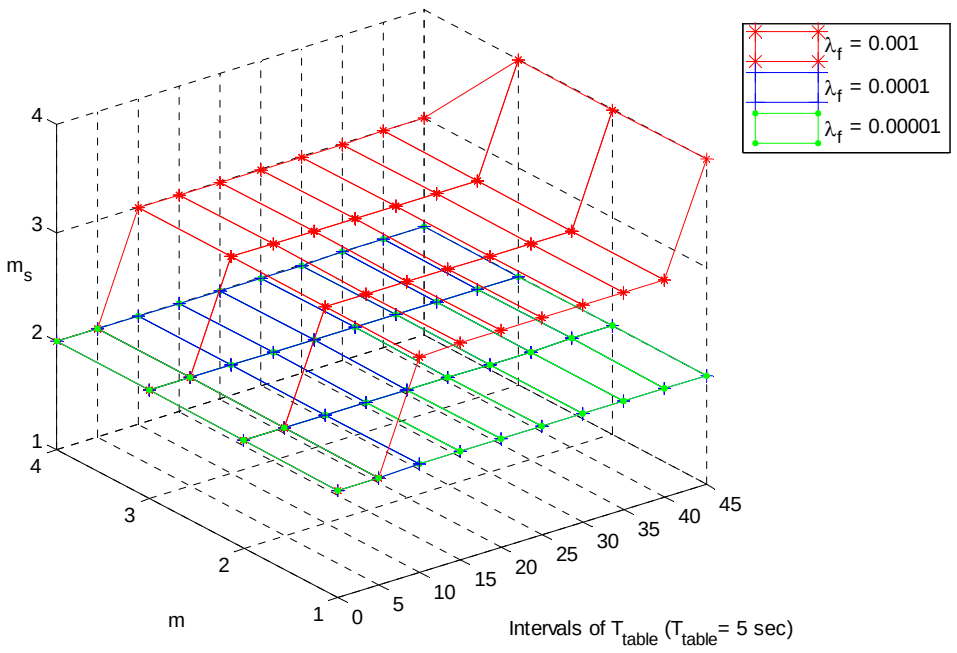

Figure 5-23: Optimal (m, m_s) vs. Time in Increment of T_{table}, with T_{table} = 5 sec, λ_f=[0.00001-0.001].

Figure 5-24 shows MTTF as a function of T_{table} and E_{table}. We observe that there exists an optimal T_{table} for each E_{table} value. When E_{table} = 0.01, the optimal T_{table} is 35 sec. When E_{table} = 0.001, the optimal T_{table} is 20 sec. When E_{table} = 0.0001, the optimal T_{table} is 5 sec. This is because when T_{table} is small, the system uses more accurate optimal (m, m_s) to trade energy off for table rebuilding. When T_{table} is large, the system uses non-optimal (m, m_s) which results in smaller MTTF values. When E_{table} is small, the system can afford to rebuild tables more frequently compared with when E_{table} is large; therefore, the optimal T_{table} interval is smaller when E_{table} is smaller. We also observe that MTTF decreases as E_{table} increases since the system spends more energy for table rebuilding as E_{table} increases.

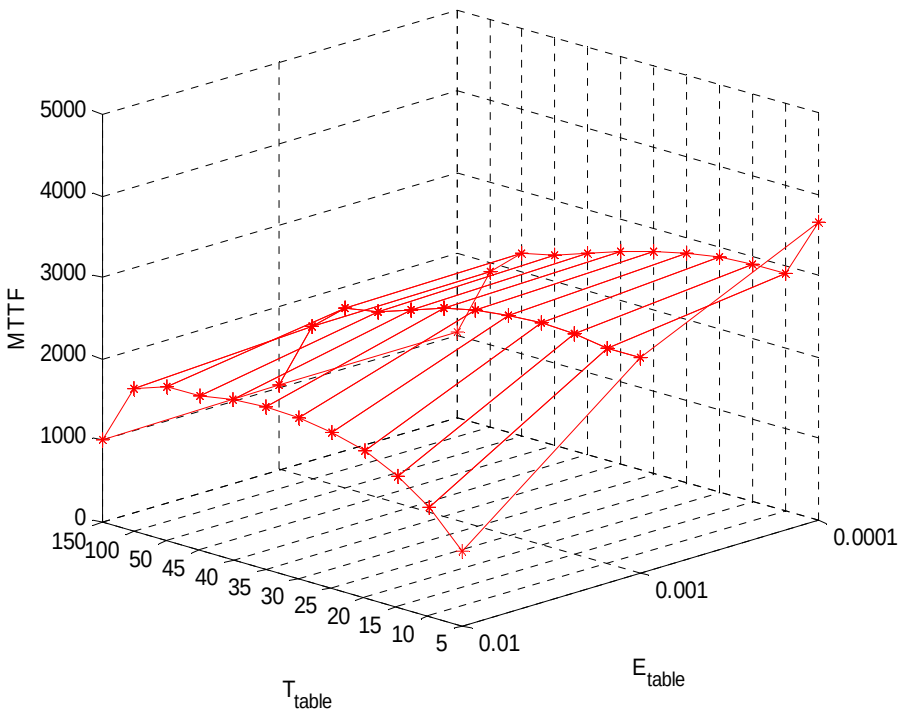

Figure 5-24: Effect of T_{table}, and E_{table} on MTTF, with λ_f =0.001, E_{table} = [0.0001 – 0.01].

Figure 5-25 shows MTTF as a function of T_{table} and λ_f. We also observe that there exists an optimal T_{table} for each λ_f value. When λ_f = 0.001, the optimal T_{table} is 20 sec. When λ_f = 0.0001, the optimal T_{table} is 25 sec. When λ_f = 0.00001, the optimal T_{table} is 100 sec. The optimal T_{table} is larger when λ_f is smaller since when λ_f is small, the optimal (m, m_s) does not change as often; therefore, the system does not need to rebuild the tables often. When λ_f is large, the optimal (m, m_s) changes more frequently; therefore, the system needs to rebuild tables more frequently which results in smaller optimal T_{table}.

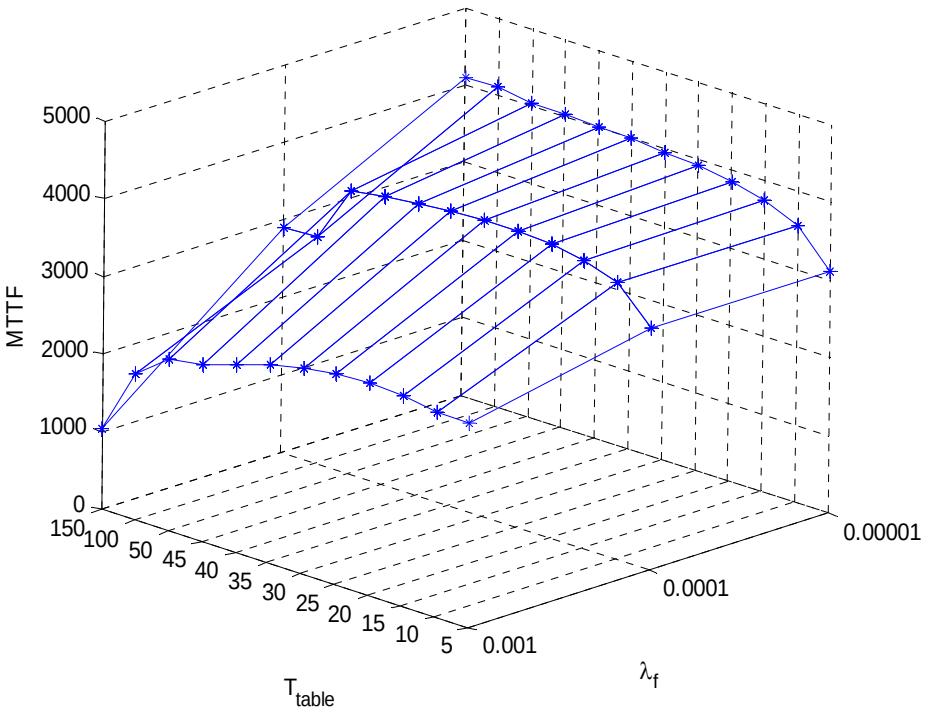

Figure 5-25: Effect of T_{table}, and λ_f on MTTF, with E_{table} =0.01 J, λ_f=[0.00001-0.001].

Chapter 6
SIMULATION

In this chapter, we first develop a simulation framework for a query based WSN based on J-Sim. Then, we describe how we simulate network dynamics including transmission and node failures, transmission speed violation, and changes in density, residual energy and radio range. We describe how we collect simulation results based on batch mean analysis to achieve statistical significance. Finally, we perform comparative analysis of simulation vs. analytical results for the purpose of simulation validation. We also conduct a series of sensitivity analysis to study the sensitivity of simulation results with respect to key model parameters.

6.1. Simulation Framework

In this section, we describe our simulation framework and methodology for conducting a simulation study. Several network simulators are available to support wireless sensor network simulation. Building on top of ns-2 [88], a discrete-event simulator, are the Monarch extension [90] to support mobile wireless simulation and SensorSim extension [91] – [92] to support sensor network simulation. Building on top of a discrete-event simulator Ptolemy [93] is VisualSense [94] that supports simulation and visualization of sensor networks. Prowler and JProwler [95] – [97] are the Matlab and Java versions that support wireless sensor network simulation used in TinyOS.

J-Sim, along with the SensorSim extension [98] – [104], is a Java open-source, component-based compositional network simulation environment gearing in simulating wireless sensor networks. Simulation study in [105] indicates that J-Sim performs better than ns-2 in terms of simulation time and is more scalable in memory usage. Existing protocol for wireless sensor networks such as LEACH, MMSPEED and GPSR were also simulated using J-Sim [106] – [110]. The SensorSim extension of J-Sim framework provides an object-oriented definition of (1) sensor nodes, sink nodes and source nodes with Poisson traffic arrival pattern, (2) wireless communication channels, and (3) physical media such as channels, propagation models, mobility models and power models (both energy producing and energy-consuming components).

Therefore, we chose to use J-Sim as the simulation framework to evaluate our proposed algorithm.

We tailor J-Sim to simulate a query-based WSN for this research. For a query-based WSN, our simulation environment consists of a Query Generator (QG) and four types of nodes: Processing Center (PC), Cluster Head (CH), Source Node (SN) and Intermediate Sensor Node (SN_I). The Query Generator acts as the user to generate queries. The PC nodes are special CHs that receive queries. A PC accepts a query from the Query Generator and sends it to the source CH. A CH chooses SNs in its cluster to perform sensor reading and relays data back to the PC. A chosen SN performs sensor reading and reports data back to the CH. Intermediate SNs relay data between the sensing SNs and the CH, and between the CH and the PC, based on our hop-by-hop data delivery scheme developed in the dissertation research. Our query-base WSN environment is illustrated in Figure 6.1.

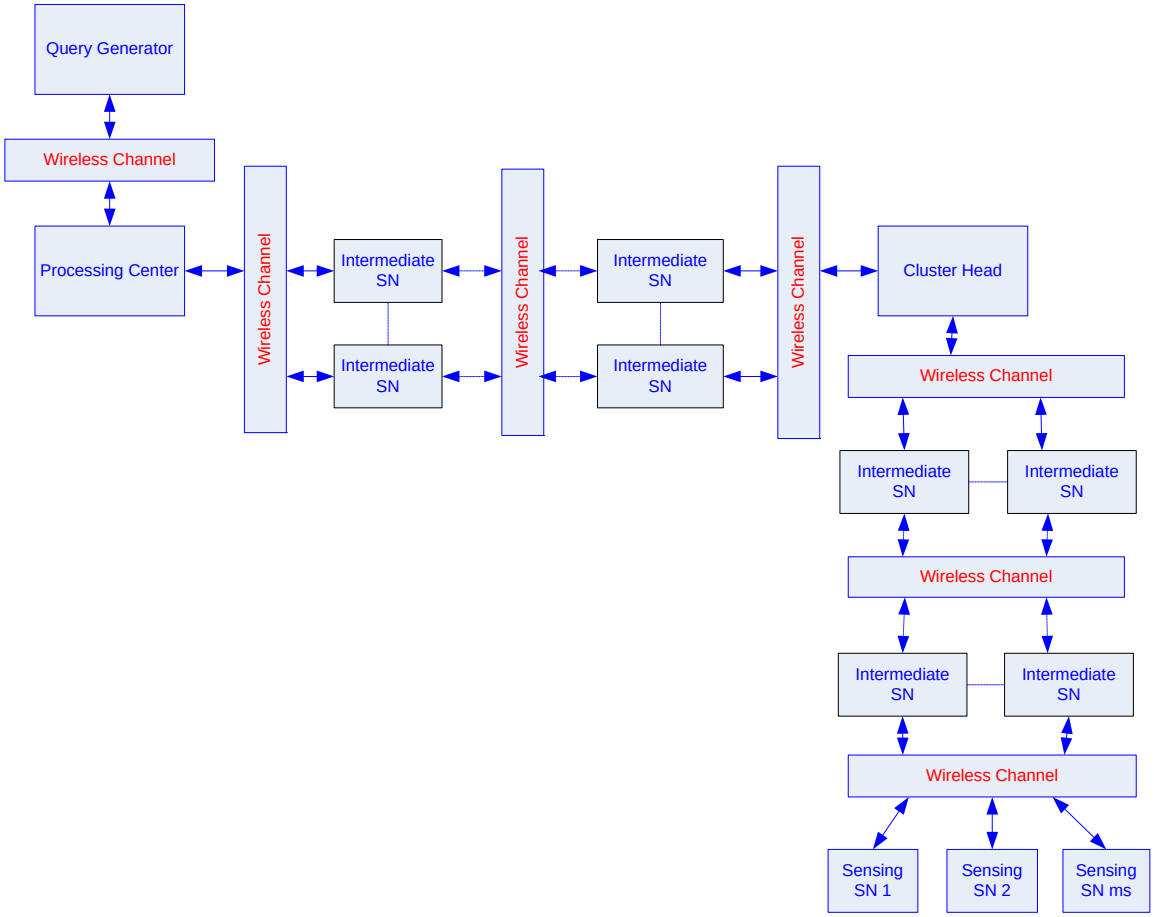

Figure 6-1: Query-based WSN Environment with HHDD Protocol.

6.2. Simulation Environment

In our simulation environment, SNs are distributed in a square terrain area of size A^2 in accordance with a population distribution function. We consider two population distribution functions, uniform distribution vs. homogeneous Poisson, and analyze the sensitivity of simulation results with respect to population distributions. To simulate a Poisson distribution, we place the first SN in the center of the square area. We then generate a random variate ("area") using an exponential random variate generator with rate of λ and use this variate generated as a radius. We create a circle with this radius from the first SN and place the next SN randomly on this circle. We repeat this process using the newly generated SN as the basis to generate the next SN until the WSN is populated with the target population.

Each SN is implemented with an application, network, link, MAC and wireless physical layers. All SNs are identical with an initial energy of E_o. Each SN is implemented with an *Energy Model* that keeps track of its remaining energy level. Both forward traffic and reverse traffic are simulated. We only simulate AFTQC without acknowledgement.

SNs use stateless non-deterministic geographic routing as described in [48]. To simulate geographic routing, we utilize the *Node Position Tracker* implemented in J-Sim. The node position tracker keeps track of the locations of all SNs. In the simulation, a SN knows its own location along with the locations of its neighbors within the radio range. Each SN determines a neighbor set consisting of nodes that are within its radio range. From the neighboring set, a forwarding set is determined, consisting of nodes that are closer to the destination. The forwarding set is divided in two groups. One group contains nodes that have progressive speed higher than the minimum requirement. The other contains nodes that have progressive speed lower than the minimum requirement. A node in the first group that has the distance closest to the radio range is chosen as the forwarding node. This is done to reduce the number of hops toward the destination and keep it as close as possible to our analytical model. In the case when data is relayed from a CH, m forwarding nodes (if available) are chosen to form m paths. If there is less than m forwarding nodes, all nodes in the first group will be chosen. If a SN is failed due to hardware failure or energy depletion, it is marked as "dead" and will not be chosen as a forwarding neighbor.

To determine the speed requirement, each node periodically exchanges its status information with its neighbors regarding location and transmission delay (d_j) to allow the progressive speed S_{jk} to be calculated dynamically based on the following equation:

$$S_{j,k} = \frac{(dist_{j,d} - dist_{k,d})}{d_j} \qquad (74)$$

where $dist_{j,d}$ is the geographic distance along the virtual direct line between node j and the destination and $dist_{k,d}$ is the geographic distance along the virtual direct line between the neighbor node k and the destination.

We use S-MAC protocol [111] – [112] in our simulation of the sensor MAC layer. In S-MAC, energy conservation and self-configuration are the primary goals, while per-node fairness and latency are less important. For collision and overhearing avoidance, S-MAC adopts a contention based scheme similar to 802.11, including both virtual and physical carrier sense and RTS/CTS exchange. The difference between S-MAC and 802.11 is that S-MAC tries to avoid overhearing by letting interfering nodes go to sleep after they hear a RTS or CTS packet. To reduce the control message overhead, S-MAC fragments a long message into many small fragments, and transmits them in burst. Only one RTS packet and one CTS packet are used to reserve the medium for transmitting all the fragments. If a neighboring node hears a RTS or CTS packet, it will go to sleep for a period of time. Another difference between S-MAC and 802.11 is that 802.11 only reserves the medium for the first data fragment and the first ACK; therefore, it has to keep listening until all the fragments are sent. To reduce the effect of idle listening, S-MAC uses periodic listen and sleep cycles. For query processing with real-time deadlines, since latency is an important factor, our implementation of the S-MAC protocol allows sensors in sleep to have the RF radio block acting as the query detector as specified in the system model.

Table 6-1 lists the main parameters used for S-MAC implementation. Most of the parameters are similar to those used in the published implementations of S-MAC with Rene and Mica motes [113] – [115]. The differences between our implementation and the published ones are that we use a bandwidth of 200Kbps as opposed to 20Kbps, a fixed duty cycle of 50% as opposed to a configurable duty cycle, and a listen interval of 300 msec as opposed to 115 msec. Another difference is that we tailor the implementation for the query-based WSNs by allowing sensors in the sleep mode to use the RF radio block as a radio detector. Each SN has an energy

module created when the simulation starts. This energy module assigns an initial energy level to the SN. Each time a radio changes states, the energy consumed will be updated. As we have described in the system model, the energy consumed by the radio detector is ignored since it consumes very little energy.

Table 6-1: S-MAC Parameters.

Parameter	**Value**
Bandwidth	**200Kbps**
Control packet length	**10 bytes**
Data packet length	**50 bytes**
MAC header length	**8 bytes**
Duty cycle	**50%**
Duration of listen interval	**300 msec**
Contention window for SYNC	**15 slots**
Contention window for data	**31 slots**

6.2.1. Query Processing

a) Query Generator

We simulate a Query Generator to perform the following functions:

- Generate queries. To implement the Query Generator, we utilized the Traffic Poisson component that allows us to generate queries following the Poisson traffic pattern with rate λ_q.
- Generate reliability and timeliness requirements (R_{req}, T_{req}) for each query based on the QoS class;
- Randomly choose a CH to be the PC to initiate the processing of the query;
- Randomly choose the source CH to respond to the query;
- If a response is returned successfully within the timeliness requirement, then increment the query count and go back to step 1; else signal system failure.

b) Source Node

A Source Node performs the following functions:

- Generate sensor data;
- Relay data to its CH. We simulate the HHDD protocol for relaying data from a SN to the source CH as described in the algorithm implementation in Chapter 3.

c) Intermediate SN

An intermediate SN, if chosen to forward data in a broadcast packet received, forwards data toward the specified destination based on stateless non-deterministic geographical routing.

d) Processing Center

A Processing Center performs the following functions:

- Estimate the transmission failure probability (e_j) for SNs located between the PC and the source CH, as well as for SNs between the source CH and source SNs within a cluster.
- Perform a lookup into the MMS table to determine the optimal (m, m_s) to use based on the estimated transmission failure probability e_j and the required deadline T_{req};
- Forward the query with the optimal redundancy (m, m_s) to the source CH and wait for replies.

e) Cluster Head

A CH performs the following functions:

- Randomly choose m_s SNs in its cluster to perform sensor reading;
- Receive replies from the selected SNs and forward the first one received to the PC; if software faults are to be tolerated, perform a majority voting before forwarding;
- Relay data to the PC according to the HHDD protocol. In the broadcast packet, specify m SNs (if available) on the first hop along the direction of the PC that satisfy the speed requirement so that these m SNs will continue to forward the packet toward the PC.

6.2.2. Status Reporting

In the simulation study, we implement both proactive and reactive approaches for status reporting. In the proactive approach, every T_{beacon}, SNs periodically inform the CH their residual energy level and local channel and transmission delay conditions as part of the clustering algorithm execution. CHs compute the average, estimated transmission failure probability e_j and transmission speed violation probability $Q_{t,jk}$ within the cluster and stores the information in an *intra-cluster* channel/delay table. CHs also exchange periodically with other CHs on channel and transmission delay conditions in their clusters. Each CH stores this summary information regarding e_j and $Q_{t,jk}$ in an *inter-cluster* channel/delay table in order to determine optimal (m, m_s) for query processing. In the reactive approach, the PC sends a request to the source CH and other CHs between itself and the source CH for information on local channel and transmission delay

conditions. Upon receiving the request from the PC, CHs send an update packet summarizing the current average e_j and $Q_{t,jk}$ values in their clusters to the PC. The PC then uses this information to determine the optimal (m, m_s) for query processing.

6.3. Simulating Network Dynamics

To model network dynamics, we simulate network conditions including transmission failure, transmission speed violation, node failure, residual energy, density change, and radio range change.

To simulate transmission failure, we assign a transmission failure probability e to each link. When a link is chosen to send a packet, we randomly generate a number between 0 and 1 and compare with the transmission failure probability. If this number is less than the transmission failure probability, the link is considered broken. If that link is the only link, the path is considered failed. For the reverse traffic, on the first hop, if m links are available, the transmission is allowed to go through the m links. If less than m links are available on the first hop, the transmission is still allowed. On subsequent hops, we only chose one link. In the case that all paths fail, the query fails and the simulation will stop with the result recorded.

To determine if we have a transmission speed violation, we calculate the actual time a packet is delivered from the source to the sink to see if it satisfies the timeliness requirement. If this time is less than T_{req}, the query is counted as a success; otherwise it is counted as a failure. If there is no node within the radio range or there is no node in the forwarding set that satisfies the speed requirement to forward the packet, then the path is considered failed. However the query is still alive as long as one path exists that is able to forward data to the PC.

To simulate node failure, we assign a failure rate λ_f to each node. Every node will have a failure probability of $q(t) = 1 - e^{-\lambda_f t}$. In order to determine how many nodes are still alive, we randomly generate a number between 0 and 1 to see if a node has failed. If this number is greater than the node failure probability $q(t)$, then the node is still alive, otherwise, the node has failed. If a node has failed, we label it as failed.

For energy, we keep track of energy whenever a node is involved in transmitting or receiving a packet. If the energy level of a node becomes less than E_T which is the energy required for transmitting, the SN is considered failed due to energy depletion. The total energy is the sum of energy of all nodes that are alive. We keep track of all sensors, their status

(alive/failed) and remaining energy. At a particular point in the simulation, the density is calculated as the number of nodes that are alive per square unit.

For transmission radius r, we periodically recalculate r based on Equation (56) and dynamically adjust the transmission radius. Once the transmission radius is adjusted, the energy consumed for transmitting or receiving is also adjusted accordingly. To deal with network dynamics, we use lookup tables (best MMS tables) to determine the optimal (m, m_s) values to use for query processing during each T_{table} interval as time progresses. These tables are pre-generated with optimal (m, m_s) at T_{table} intervals based on $q(t)$, $n(t)$, $\lambda(t)$, $r(t)$ and $E_0(t)$ at that time point.

For the proactive approach, each CH periodically (after each T_{table} period) sends a packet to other CHs carrying the information of average (e_j, $Q_{t,jk}$). This information will be stored in the *inter-cluster channel/delay* table. When a query arriving at the CH, a table lookup into its *inter-cluster channel/delay* table by the PC is performed to retrieve the average (e_j, $Q_{t,jk}$) value out of those CHs located between the PC and the source CH. These average (e_j, $Q_{t,jk}$) values then can be used as indexes into the best MMS table to lookup for the optimal level of redundancy (m, m_s) that should be used for query processing.

For the reactive approach, The PC will send an inquiry packet to the source CH as well as all CHs located between the PC and the source CH to request an update on the average (e_j, $Q_{t,jk}$) of SNs in their clusters. Upon receiving the request from the PC, CHs will send back a reply packet carrying the average (e_j, $Q_{t,jk}$) to the PC. These average (e_j, $Q_{t,jk}$) values then can be used as indexes into the best MMS table to lookup for the optimal level of redundancy (m, m_s) that should be used for query processing.

6.4. Simulation Processing

A query is considered as not being executed successfully if one of the following conditions happens:

- If all m_s SNs fail to deliver sensor readings to the source CH, due to a combination of link failure, SN energy depletion or SN hardware/software failures.
- All paths between the CH and the PC are broken, due to a combination of link failure, SN energy depletion and SN hardware failure.
- The query result is not returned within the deadline requirement T_q. We accumulate the time it takes to propagate the results back based on the progressive speeds of the SNs

chosen to forward data. For each segment (from a SN to the CH and from the CH to the PC) we use the transmission time of the first path that returns the query result to get the total response time. If SN measurement software faults are considered, the transmission time for all the SN-CH paths to return sensor readings to the CH is considered instead.

The simulation runs in rounds. In each round, we record the number of queries processed successfully, which is recorded as instance of the system MTTF. R_q is computed by the ratio of the number of queries that do not fail due to SN hardware/software or channel failures over the total number of queries. E_q is computed by the average energy consumed per query over all queries that do not fail.

We have developed a Statistical Analyzer module to compute the MTTF value with statistical significance. The Statistical Analyzer uses *batch mean analysis* to obtain MTTF, treating each MTTF obtained from a simulation run as a data point in order to obtain the average MTTF within a specified confidence interval and accuracy. We run the simulation until we archive 95% confidence level and 10% accuracy. To achieve this, we collect observations in batches with 1000 observations in each batch. In one batch we obtain a batch mean out of 1000 observations collected. We run at least 10 batches to get a minimum of 10 batch means from which we calculate the grand mean and estimate the difference of the grand mean from the true mean with 95% confidence. If the accuracy obtained is greater than 10%, we run more batches and collect more observations until the specified 10% accuracy requirement is met. We run the simulation for the optimal (m, m_s) and other non-optimal (m, m_s) values. The results are used to draw a 3-D graph representing MTTF based on m and m_s against which analytical results are compared and validated. R_q and E_q computed are also compared against analytical results. The difference between analytical results vs. simulation results obtained is represented by the mean percentage difference of mismatch and the associated standard deviation. To obtain the mean percentage difference, we first calculate the percentage difference between each MTTF value pair under the same set of environment parameters. Then we calculate the mean of the percentage differences. The standard deviation is calculated based on the absolute differences between MTTF value pairs.

6.5. Simulation Results

In this section, we show simulation results to compare with analytical results for the purpose of simulation validation. Table 6.2 lists a set of parameters along with their default values used in the simulation.

Table 6-2: Default Parameter Values Used in the Simulation Study.

Parameter	**Value**
m	**[1 – 4]**
m_s	**[1 – 4]**
N	**600**
n_s	**100**
q	**0.0001**
e	**0.0001**
r	**40 m**
f	**½**
λ_q	**1 query/sec**
A	**400m**
n_b	**50 bytes**
n_q	**10 bytes**
E_T	**0.0000264 J**
E_R	**0.00002 J**
E_o	**0.05 J**
$E^s_{threshold}$	**0.0000264 J**
$T_{clustering}$	**5 sec – 5 min**
$T_{\mathbf{req}}$	**1.0 sec**
$T_{\mathbf{beacon}}$	**10 sec**
B	**200Kb/s**
T_{table}	**5 sec – 150 sec**
E_{table}	**0.0001 – 0.01 J**
λ_f	**0.00001 – 0.001**

We use a smaller number of sensor nodes (n = 600) and a small value of initial energy E_o (0.05 J) to keep the simulation time within a reasonable amount of time in a machine with the following specification: 2.4 GHz, dual core processor with 2GB of memory. Below we compare

simulation results obtained with analytical results under identical parameter value sets. In general, the simulation results correlate well with the analytical results.

In the numerical analysis, the query arrival rate (λ_q) and the sensor density λ are fixed parameter values. In the simulation, the inter-arrival time between queries is exponentially distributed with the average being $1/\lambda_q$. The sensor density λ is dynamically calculated based on the number of sensor nodes that are alive. In the numerical analysis, MTTF values are calculated based on Equations (48) and (61). In the simulation, MTTF values are obtained by observations using the batch mean analysis method. In both numerical analysis and simulation, the table rebuilding interval (T_{table}) and the energy for table rebuilding (E_{table}) are chosen from a range of discrete parameter values.

In the numerical analysis, the transmission failure probability (e_j) is a fixed parameter value. Since the progress speed S_{jk} is not known until run time, we calculate the transmission speed violation probability $Q_{t,jk}$ based on Equation (19) assuming that S_{jk} is uniformly distributed between a range [a, b]. In the simulation, the transmission delay and transmission failure probability are collected and updated by the sensor MAC layer. The progress speed S_{jk} is determined based on Equation [74]. The transmission speed violation probability $Q_{t,jk}$ is calculated dynamically. If S_{jk} is above S_{req} then $Q_{t,jk} = 0$; otherwise, $Q_{t,jk} = 1$, where S_{req} is the minimum speed requirement calculated based on Equation (15) for the forward traffic and based on Equation (16) for the reverse traffic.

In the numerical analysis, the probability of a SN becoming a CH, p, is a fixed parameter value and is determined by $1/n_s$ where n_s is the number of sensor nodes in the cluster. In the simulation, we implement a clustering algorithm to simulate cluster formation and cluster head rotation. SNs are selected to be CHs based on the cluster head selection algorithm described in Chapter 2 and they communicate with each other via multihops. The clustering algorithm is executed periodically in every $T_{clustering}$ interval and the probability of becoming a CH is dynamically calculated.

Figure 6-2 compares simulation results obtained vs. analytical results for a query-based WSN operating under the set of parameter values listed in Table 6.1. The MTTF obtained considers both the forward and reverse traffics. For this simulation, the mean percentage difference in MTTF between analytical and simulation results is 5.93%, with the standard deviation being 90.1. As a reference, the analytical MTTF value at the optimal (m, m_s) is 5323.68

and the simulation MTTF at the optimal (m, m_s) is 4896.41. We conclude that the simulation and analytical MTTF curves correlate well, with the same optimal (m, m_s) at (2, 2).

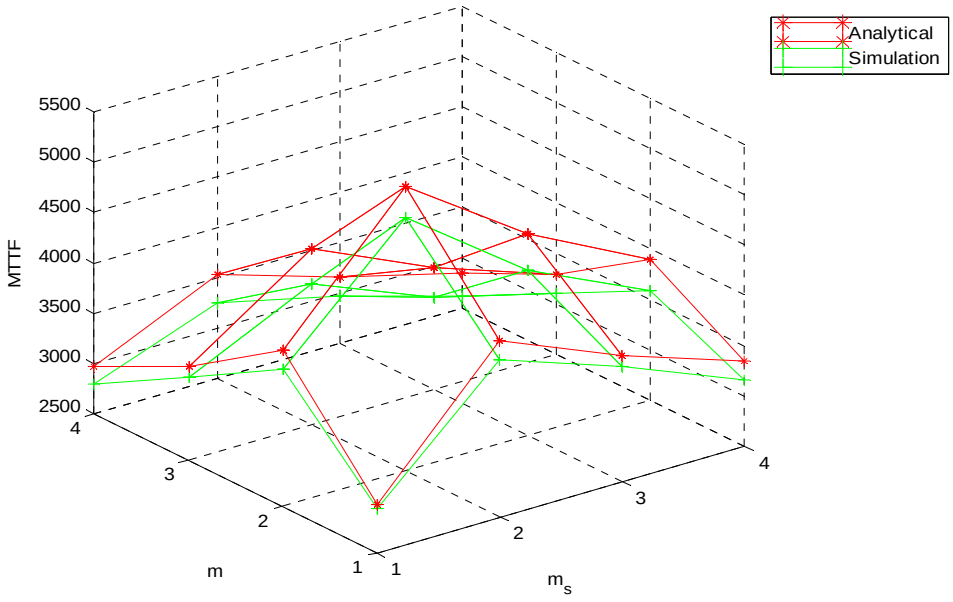

Figure 6-2: Comparison of Analytical and Simulation Results for MTTF vs. (m, m_s).

Figure 6-3 compares simulation results obtained vs. analytical results for the average energy used per query E_q. For this simulation, the mean percentage difference in E_q between analytical and simulation results is 6.83% and the standard deviation is 2.49e-4. Figure 6-4 compares simulation results obtained vs. analytical results for the query reliability R_q. For this simulation, the mean percentage difference in R_q between analytical and simulation results is 0.00625% and the standard deviation is 2.17e-5. We conclude that the simulation results show good correlation with the analytical results, with the same trend exhibited.

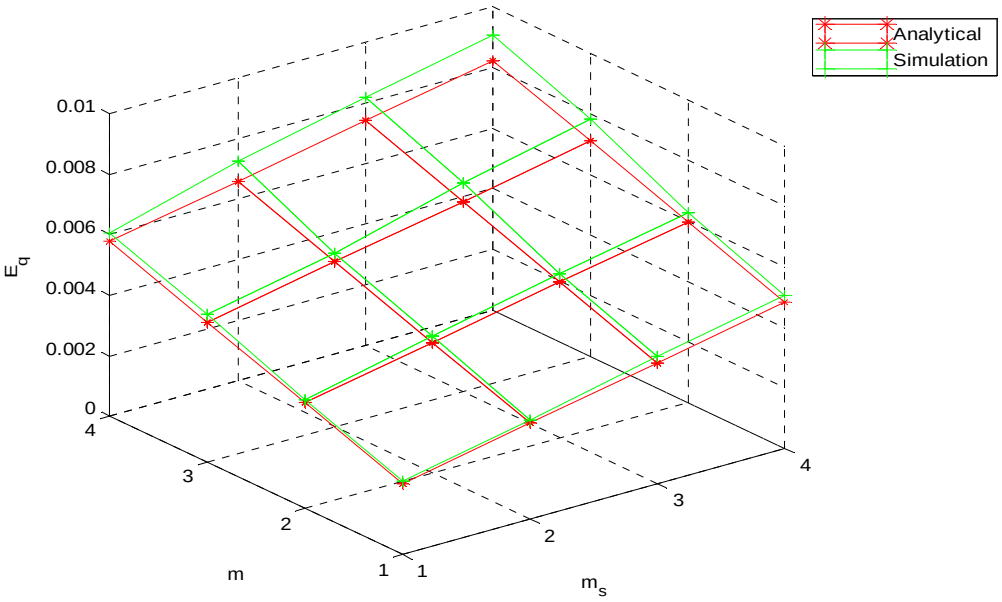

Figure 6-3: Comparison of Analytical and Simulation Results for Average Energy Consumed Per Query.

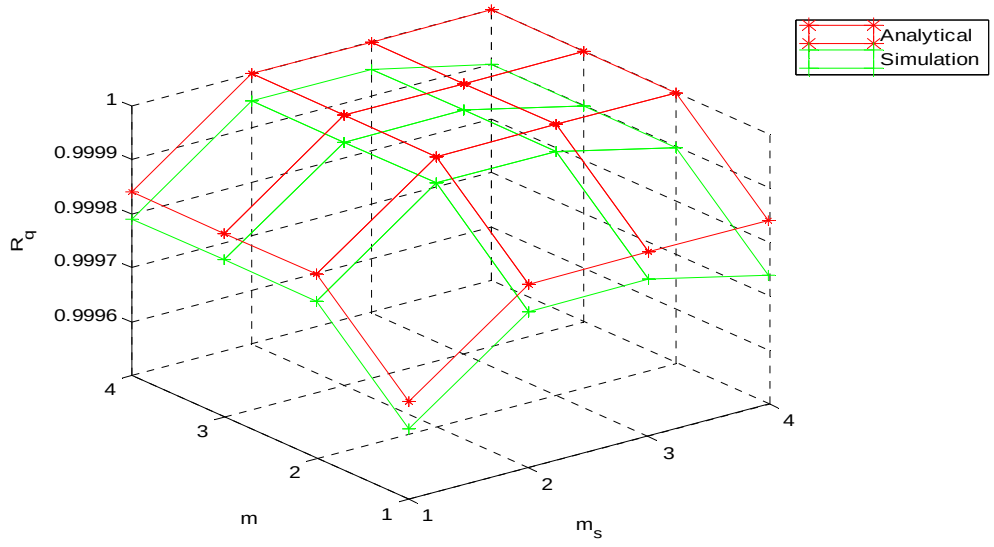

Figure 6-4: Comparison of Analytical and Simulation Results for Query Reliability.

We have also conducted a simulation study that considers changes in the network conditions. Specifically, in addition to simulating transmission failure and transmission speed

violation, we also simulate sensor node hardware failure and energy depletion as described in Section 6.3. Figure 6-5 compares simulation results vs. analytical results when such network dynamics are considered. In this simulation, the mean percentage difference is 3.89% and the standard deviation is 65.56. The analytical MTTF value at the optimal (m, m_s) is 5323.68 and the simulation MTTF value is 5110.4. Again the results show good correlation. Both simulation and analytical results confirm that in a WSN characterized by the set of parameter values in Table 6.1, the system can better tolerate sensor failures due to hardware or energy depletion when more paths are used.

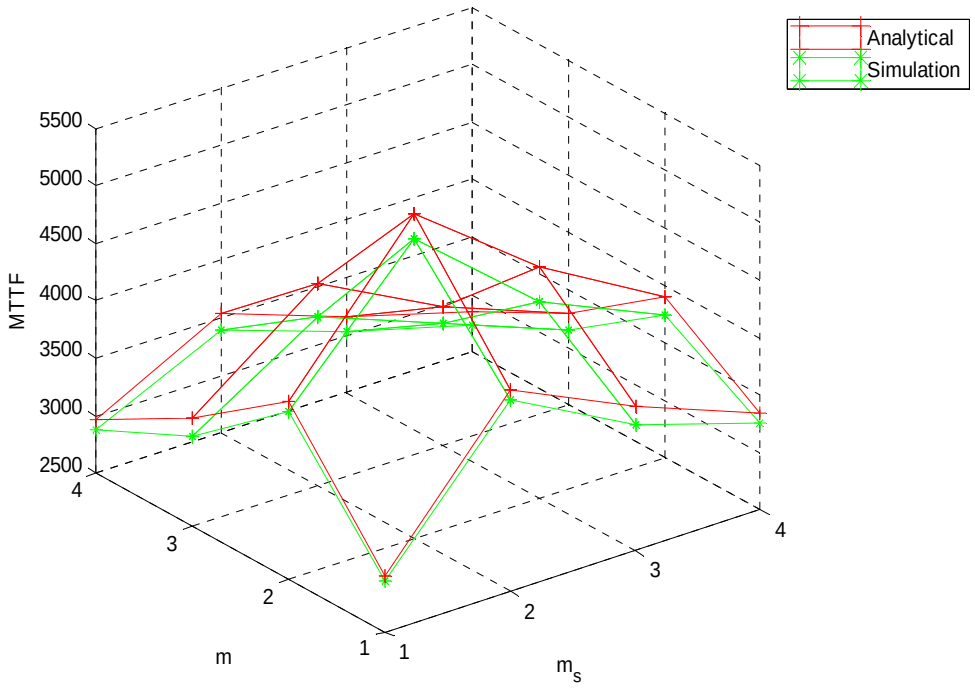

Figure 6-5: Comparison of Analytical and Simulation Results in MTTF vs. (m, m_s) when Hardware Failure and Energy Depletion are Considered.

We have conducted simulation to validate the analytical results presented in Section 5.3 for the effect of clustering intervals on MTTF. Figure 6-7 shows the simulation results, which correlate well with the analytical results shown in Figure 5-9. Both simulation and analytical results confirm that using a short clustering interval will result in a smaller MTTF since more energy would be consumed when the clustering algorithm is executed more often. Under the workload specified, simulation results show that the system achieves a perfect rotation when

$T_{clustering}$ = 5 sec. and near-perfect rotation when $T_{clustering}$ = 20 sec. The MTTF achievable under $T_{clustering}$ = 20 sec. is about the same as that under a higher clustering interval under $T_{clustering}$ = 5 min, so we conclude that the assumption of perfect rotation in the analytical model is justified.

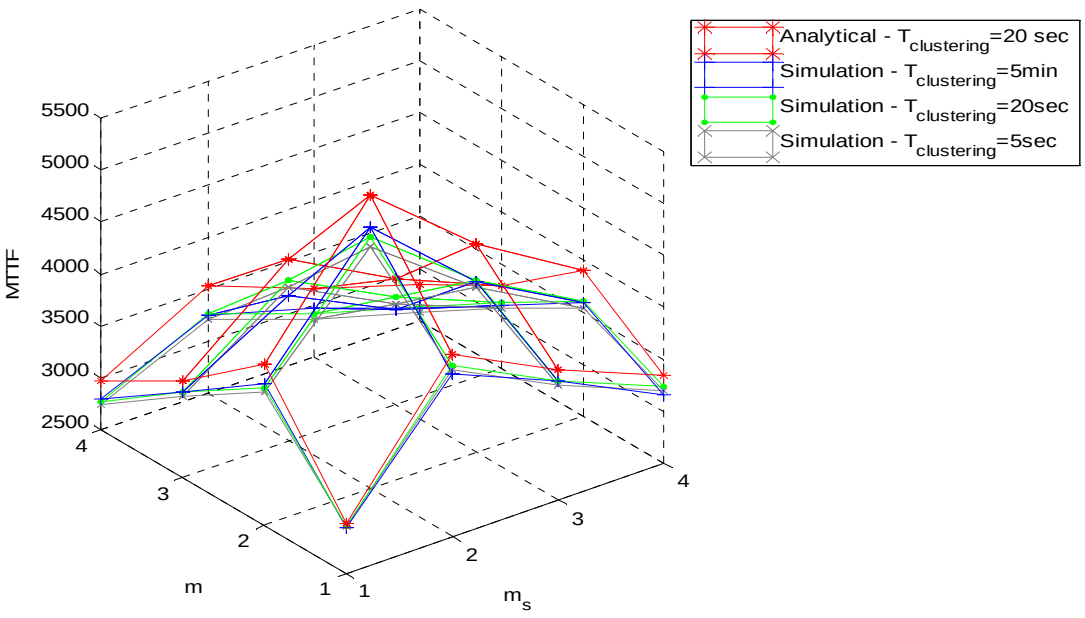

Figure 6-6: Simulation Results for the Effect of Clustering Intervals on MTTF vs. (m, m_s).

We have conducted simulation studies to compare the case when SNs are distributed according to a homogeneous Poisson process vs. the case when SNs are distributed uniformly. Figure 6-7 shows that the simulation results are insensitive to these two types of distribution used. For this simulation, the mean percentage difference is 0.69% and the standard deviation is 43.63. The MTTF value at the optimal (m, m_s) for the case of uniformly distribution is 4896.41 and the MTTF value for the case of Poisson distribution is 4945.37.

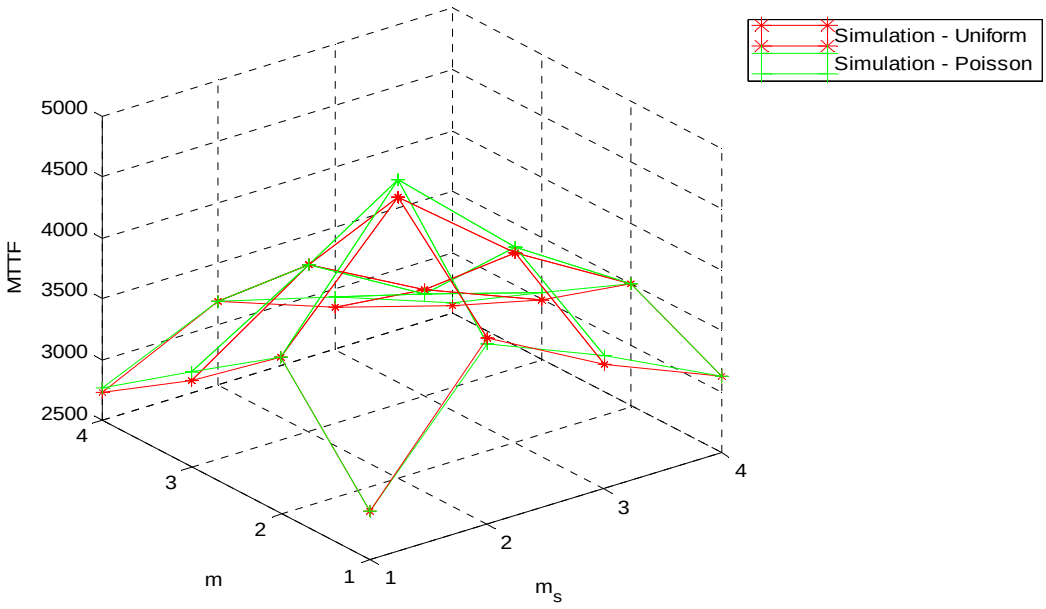

Figure 6-7: Comparison of Simulation Results between Poisson and Uniform Distribution of Sensor Nodes.

Next, we conduct simulation studies to compare Proactive vs. Reactive approaches for status exchange. Figure 6-8 compares the two approaches for the case when the query arrival rate is low (λ_q = 1). The mean percentage difference in MTTF between these two approaches is 0.8% and the standard deviation is 28.36. The simulation results show that when the query arrival rate is low, the Reactive approach performs better than the Proactive approach. This correlates with the analytical results shown in figure 5-17. Figure 6-9 compares the two approaches for the case when the query arrival rate is high (λ_q = 5). The mean percentage difference in MTTF between these two approaches is 0.86% and the standard deviation is 29.9. Again, the simulation results correlates with the analytical results shown in Figure 5-18 and confirms that when the query arrival rate is high, the Proactive approach performs better than the Reactive approach.

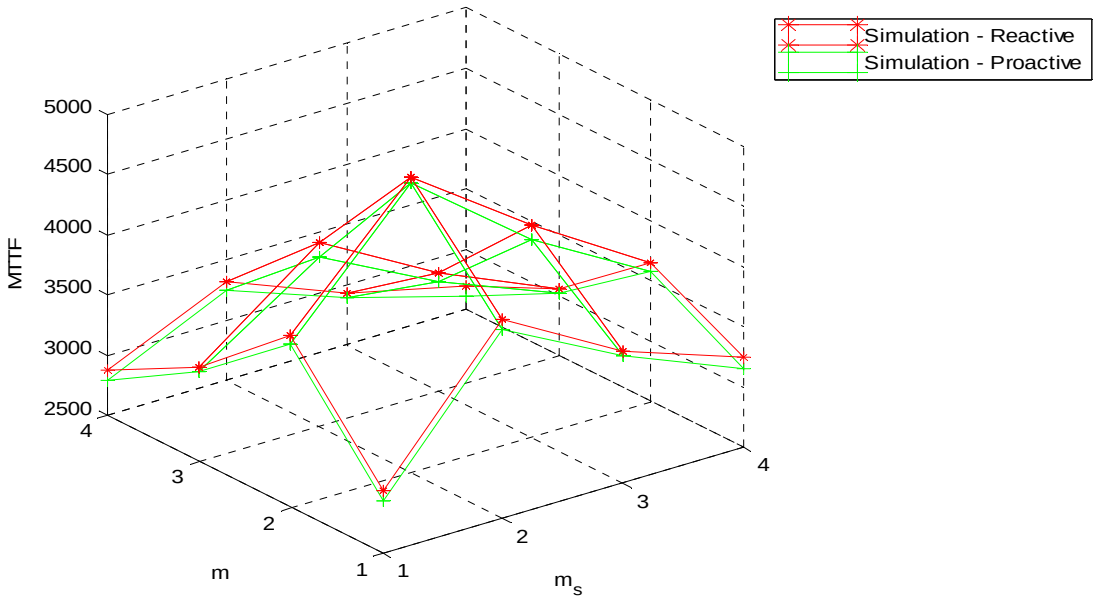

Figure 6-8: Comparison of Simulation Results between Reactive and Proactive Status Exchange for Low Query Arrival Rate ($\lambda_q = 1$).

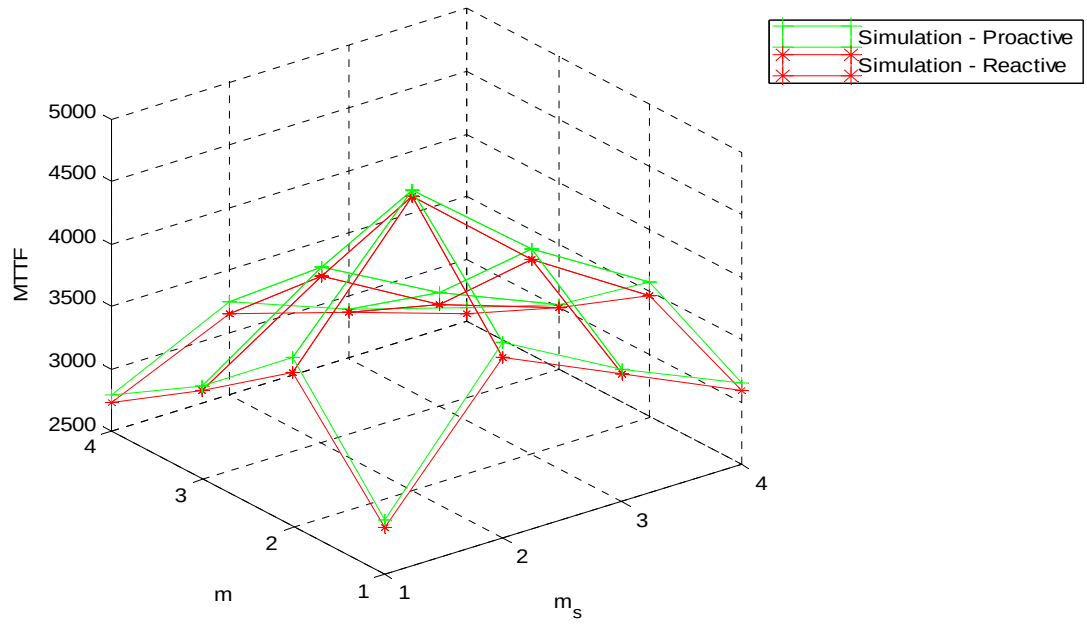

Figure 6-9: Comparison of Simulation Results between Proactive and Reactive Status Exchange for High Query Arrival Rate ($\lambda_q = 5$).

Lastly, we show the effect of network dynamics, i.e., changes in $q(t)$, $n(t)$, $r(t)$, $\lambda(t)$ and $E_0(t)$ as time progresses on MTTF. In the simulation, we use the optimal (m, m_s) tables pre-generated at T_{table} intervals to lookup for the best (m, m_s) used for query processing in each interval as time progresses. Figure 6-10 shows MTTF as a function of T_{table} and E_{table}. The simulation results correlate with the analytical results with the same optimal T_{table} for each E_{table} value. The mean percentage difference is 4.87% and the standard deviation is 44.6. The optimal T_{table} when E_{table} = 0.0001 is 5 sec. The optimal T_{table} when E_{table} = 0.001 is 20 sec. The optimal T_{table} when E_{table} = 0.01 is 35 sec.

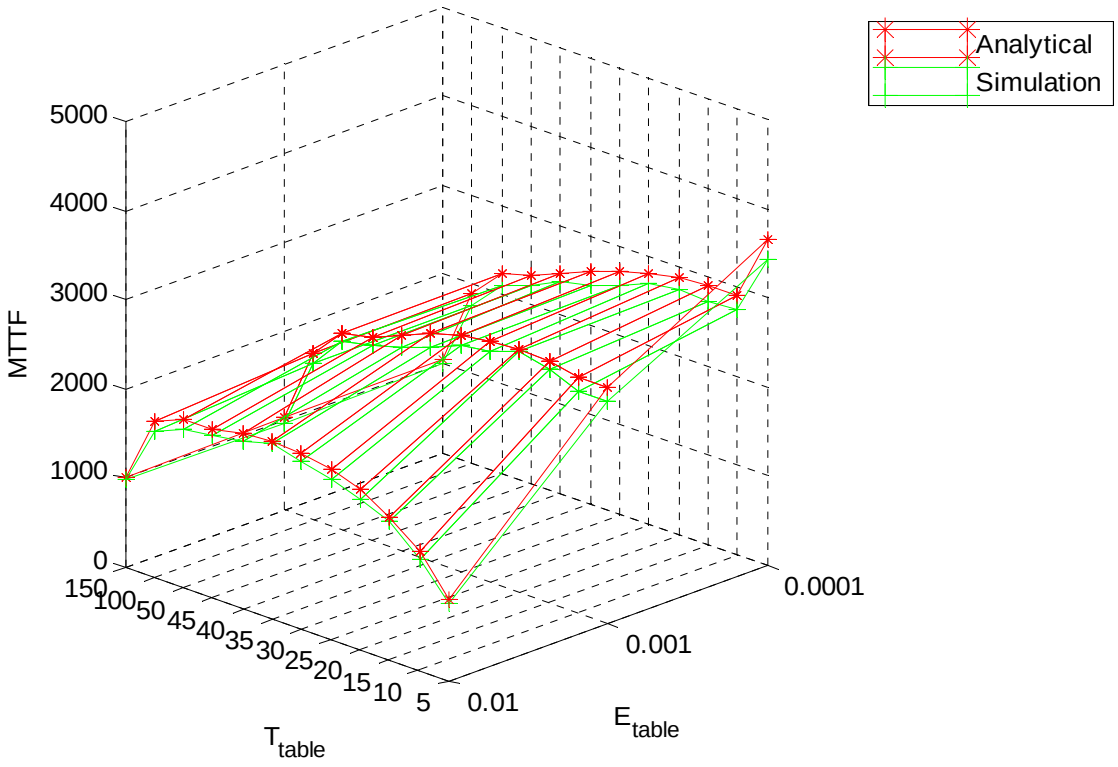

Figure 6-10: Comparison of Analytical vs. Simulation Results for the Effect of T_{table} and E_{table} on MTTF.

Figure 6-11 shows MTTF as a function of T_{table} and λ_f. The simulation results also correlate with the analytical results with the same optimal T_{table} for each λ_f value. The mean percentage difference is 4.59% and the standard deviation is 56.83. The optimal T_{table} when = 0.00001 is 100 sec. The optimal T_{table} when = 0.0001 is 25 sec. The optimal T_{table} when = 0.001 is 20 sec.

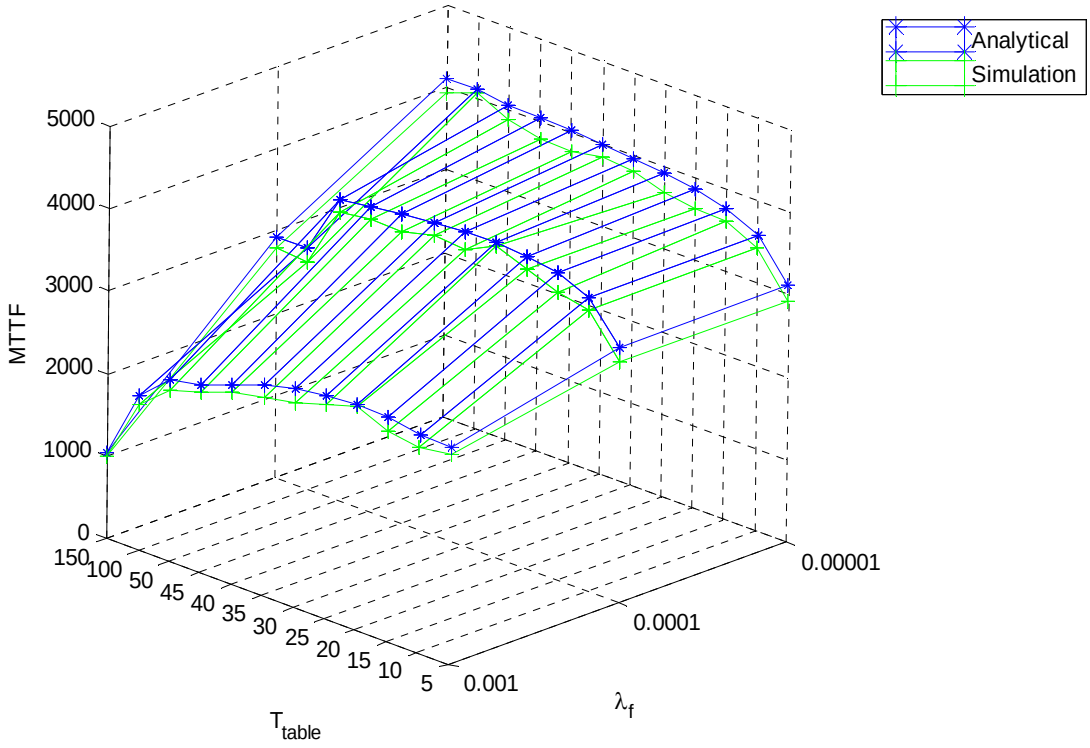

Figure 6-11: Comparison of Analytical vs. Simulation Results for the Effect of T_{table} and λ_f on MTTF.

6.6. Sensitivity Analysis

Below we conduct a sensitivity analysis of e_j (wireless channel transmission failure probability) on optimal (m, m_s) and MTTF. We also show the effect e_j on $Q_{t,jk}$ and energy consumption, and consequently, on MTTF.

Figure 6-12 shows MTTF as a function of e_j and compares analytical vs. simulation results. As expected, the results show that when e_j increases, MTTF decreases. Also MTTF decreases dramatically when e_j is relatively high. This is because when e_j is high, the system encounters more failures due to link failures. The same trend is exhibited by both analytical and simulation results, which show very good correlations.

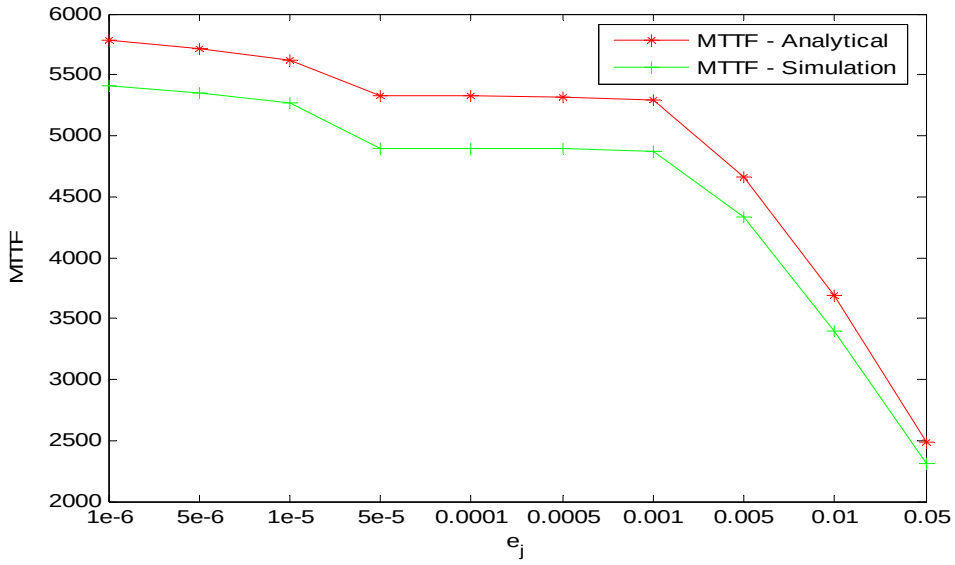

Figure 6-12: Effect of e_j on MTTF.

Figure 6-13 shows energy consumption per query as a function of e_j. The results show that when e_j increases, the energy consumed per query also increases. This is because the system requires more path redundancy to cope with link failures. This consequently decreases the system MTTF.

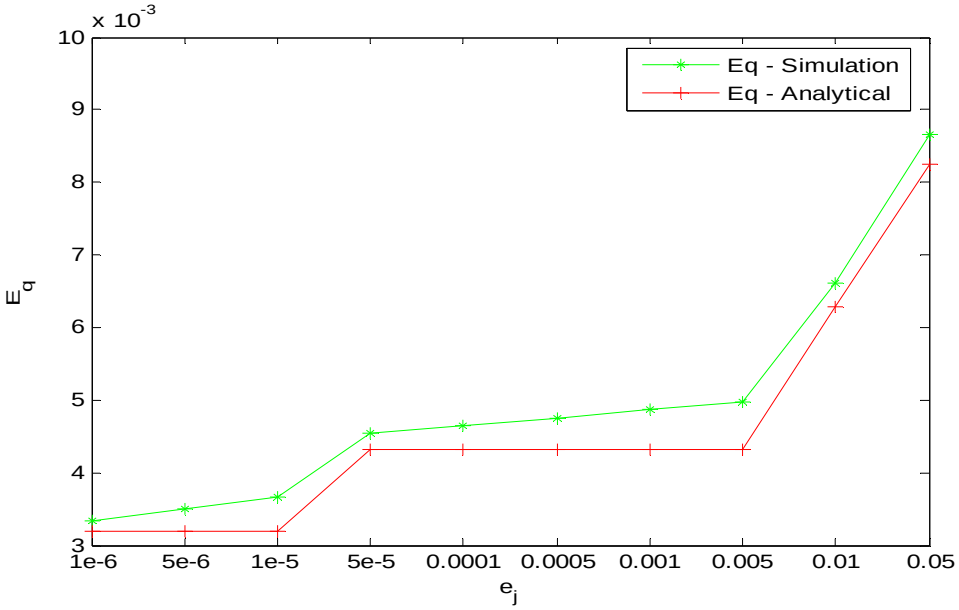

Figure 6-13: Effect of e_j on E_q.

Figure 6-14 shows the effect of e_j on the optimal number of redundant paths (m). The results show that when e_j increases, m also increases since the system requires more paths to cope with link failures. The simulation results coincide with the analytical results in the optimal number of paths used.

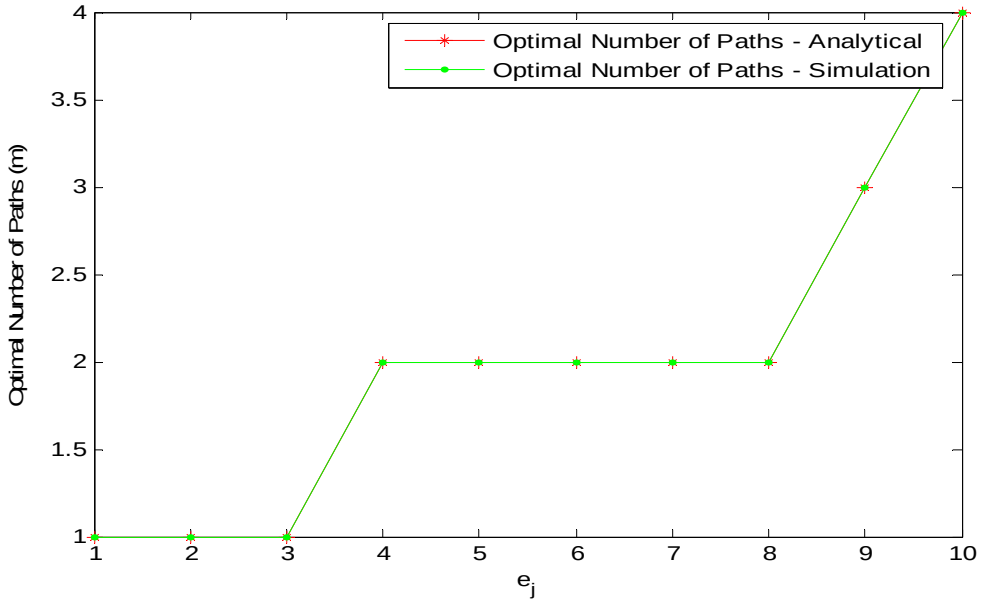

Figure 6-14: Effect of e_j on m.

Figure 6-15 shows the effect of e_j on the optimal number of source SNs (m_s). The results show that when e_j is relatively large, the number of optimal sources also increases. This is because more sources are required to create more paths between the source SNs and a source CH to cope with link failures. The simulation results coincide with the analytical results in the optimal number of source SNs used.

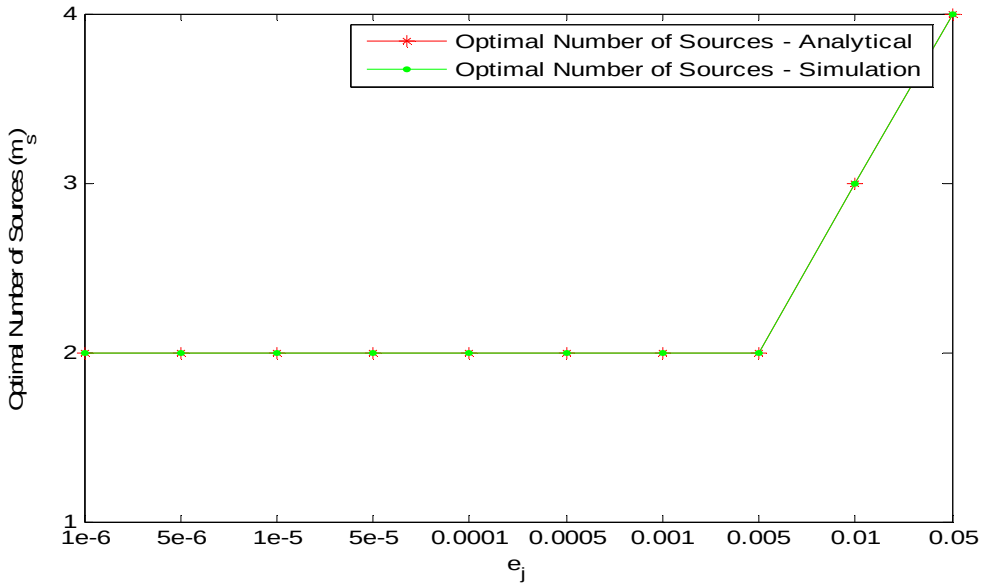

Figure 6-15: Effect of e_j on m_s.

Figure 6-16 shows the effect of e_j on $Q_{t,jk}$ via a simulation study. In the simulation, we collect the values of progressive speed $S_{j,k}$ for a number of links randomly chosen in the system to determine the link speed violation probability $Q_{t,jk}$ and get the average value of $Q_{t,jk}$. The results show that when e_j increases, $Q_{t,jk}$ also increases. This is because with higher e_j, the progressive speed is lower, and consequently, there would be less links that satisfy the speed requirement.

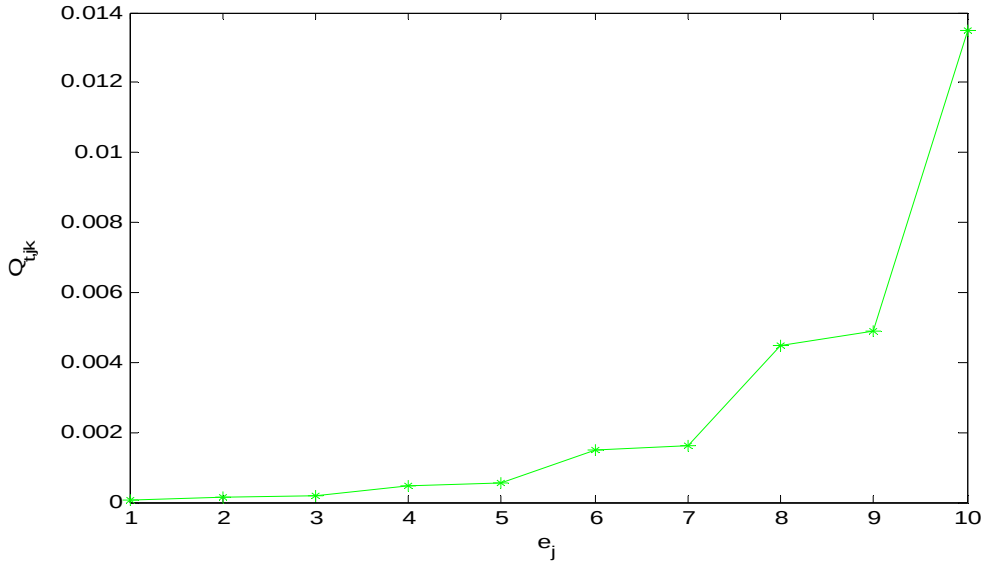

Figure 6-16: Effect of e_j on $Q_{t,jk}$.

6.7. Conclusion

We have conducted extensive simulation studies to validate analytical results obtained in Chapter 5 based on mathematical models developed in Chapter 4. Through simulation, we have studied:

- The effect of network dynamics on MTTF, including changes of node density, node population, node energy, and radio range for maintaining connectivity;
- The effect of clustering intervals on MTTF;
- The effect of proactive vs. reactive status exchange methods on MTTF.

The simulation results were collected based on the batch mean analysis method with 95% confidence interval and 10% accuracy. In general the simulation results correlate well with the analytical results, showing the same trend in MTTF vs. (m, m_s) as well as the optimal (m, m_s) set when given the same set of network and environment conditions. The mean percentage difference between simulation and analytical results in general is around 5%. It is about 1% in the best case and 7% in the worse case. We have also conducted a series of sensitivity analysis to understand the sensitivity of simulation results with respect to key model parameters. We conducted identical simulation studies based on Poisson vs. uniform population distributions and by comparison we found that simulation results obtained are quite insensitive to the type of population distribution used. We have also tested the sensitivity of $Q_{t,jk}$ and subsequently MTTF obtained with respect to e_j and concluded that e_j is an important design parameter that drastically affects MTTF. We found excellent correlation between analytical and simulation results in these sensitivity analyses.

Chapter 7
SUMMARY AND FUTURE WORK

The world is witnessing a rapid expansion in the adoption of wireless sensor networks (WSNs) deployed in applications and spaces such as smart homes, health management, public safety and tactical surveillance, emergency management, and condition-based maintenance applications on industrial, military, and space platforms. Many of these emerging applications require large-scale deployment of sensor networks and often have application-specific quality-of-service (QoS) requirements to include timeliness, reliability, availability, integrity and security. Given the large-scale of such networks, their highly dynamic topology, their resource challenged nodes with the subsequent need for node cooperation, and their likelihood of being deployed in inhospitable environments with high rate of node and link failures, provisions must be made for WSNs to operate unattended for a long period of time where sensor replacement or energy replenishment may not be feasible. Consequently, QoS-awareness intertwined with energy conservation is a core priority in WSN design.

The goal of this dissertation research is to design and build dependable, large-scale sensor networks that are resilient, long-lived networks endowed with self-adaptive QoS control capabilities to efficiently assure dynamic QoS requirements and effectively serve the emerging QoS aware applications. Our approach is to exploit the inherent redundancy in WSNs and adaptively manage redundancy to satisfy application QoS requirements while maximizing network lifetime from the application perspective. Specifically, when given a query's QoS requirements, we dynamically and adaptively identify optimal "path-level" and "source-level" redundancy so that not only QoS requirements are satisfied, but also the lifetime of the system is maximized under varying network conditions. On the one hand, uncontrolled redundancy may help satisfy reliability and timeliness requirements. On the other hand, it does not scale well and would lead to excessive energy consumption.

Our research deals with application-specific QoS requirements in terms of timeliness and reliability for queries issued from the processing center to the WSN nodes. Our approach is based on the development of a QoS-aware routing protocol with the design notion of adaptive fault tolerant QoS control (AFTQC) to achieve application-layer QoS requirements, utilizing

knowledge obtained from MAC and physical layers such as wireless channel error, transmission speed, and energy status for adaptive QoS control decision making. Specifically, we develop a hop-by-hop data delivery (HHDD) protocol to reduce energy consumption while satisfying queries QoS requirements. The HHDD protocol is embedded into AFTQC for determining optimal path and source-level redundancies during data delivery to maximize the mean time to failure of a clustered query-based WSN. We develop a theory for calculating the system MTTF to assess the effect of redundancy on system lifetime due to AFTQC and validate the theoretical predictions with extensive simulation studies.

In summary, this dissertation research has accomplished the following tasks:

1. We developed a probability model to calculate the query success probability (R_q), the energy consumption of a query (E_q), and, consequently, the system MTTF. We also considered energy consumption due to status exchange and periodic update activities to obtain current status of the system. We evaluated the effect of both forward and reversed traffics and the effect of different clustering intervals on MTTF analytically and by simulation. We also evaluated the effectiveness of proactive and reactive status exchange methods analytically and by simulation.
2. We developed an analytical solution to deal with the case when there are multiple QoS levels from multiple query classes that may query the sensor network concurrently and examined the effect of transmission interference due to concurrency. In particular, we analyzed the case in which concurrent queries may access a cluster simultaneously. We also developed solutions to deal with multiple QoS levels and analyzed their effect analytically.
3. We developed a probability model to deal with network dynamics, i.e., changes in density of nodes (λ), residual energy of nodes (E_o), and radio range (r) due to node failures and change of node connectivity. We provided a detailed analysis of the effect of network dynamics on MTTF. We validated analytical solutions with simulation studies.
4. We analyzed the use of acknowledgement and timeout mechanisms in AFTQC and identify the optimal (m, m_s) setting that minimizes MTTF, as well as conditions under which no-ACK is better than ACK-based data delivery schemes, or vise versa. We evaluated the tradeoff between the AFTQC algorithm without acknowledgement and with

acknowledgement analytically. In particular, we compared AFTQC with a baseline model in which ACK is used without redundancy and demonstrated the benefits of AFTQC.

5. We analyzed the effect of software sensor faults such as measurement faults, in addition to hardware sensor faults, on MTTF by considering fault masking based on voting (when more than two SNs are used). We evaluated the effect of software sensor fault on MTTF analytically.
6. We developed a simulation environment for a clustered WSN system using S-MAC as MAC layer protocol. The simulation environment allows us to simulate query generation and query processing based on our AFTQC algorithm. We validated analytical results obtained from the theory for the effect of clustering interval, effect of network dynamics in terms of changing node density, residual node energy and radio range, and effect of proactive vs. reactive status exchange methods on the system MTTF. In particular, we validated a clustering interval under which a near-perfect CH rotation is achievable by simulation to justify the assumption of perfect rotation among SNs for the role of a CH. In general, the simulation results correlate well with the analytical results. The mean percentage difference between simulation and analytical results in general is about 5%. It is around 1% in the best case and 7% in the worse case.
7. We conducted a series of sensitivity analysis and concluded that the results obtained are insensitive to the type of population distribution used but sensitive to e_j which drastically affects the optimal settings for maximizing MTTF in the design. We conclude that e_j must be obtained at runtime through cross-layer designs and be used as a parameter for dynamic table lookup to determine the best path and source redundancy used for maximizing MTTF.

The dissertation research has resulted in the following publications:

- Anh Phan Speer and Ing-Ray Chen, "Effect of Redundancy on the Mean Time to Failure of Data Sensors Systems," *20th IEEE International Conference on Advanced Information Networking and Applications* (AINA 2006), Vienna University of Technology, Vienna, Austria, Vol. 2, April 18-20, 2006.
- Anh Phan Speer and Ing-Ray Chen, "On Optimal Path and Source Redundancy for Achieving QoS and Maximizing Lifetime of Query-Based Wireless Sensor Networks," *IEEE*

14th International Conference on Measurement and Simulation of Computer and Telecommunication Systems (MASCOTS '06), Monterey, CA, September 11-13, 2006.

- Anh Phan Speer and Ing-Ray Chen, "Effect of Redundancy on the Mean Time to Failure of Cluster-based Data Sensors Networks," *International Journal of Computation and Concurrency: Practice and Experience*, Vol. 19, 2007, pp. 1119-1128.
- Ing-Ray Chen and Anh Phan Speer, "On Optimal Redundancy for Maximizing System Lifetime of Query-Based Wireless Sensor Networks," submitted to *IEEE Transactions on Dependable and Secure Computing*, Feb. 2008.

Some opportunities for future research from this dissertation are: (1) extending the design, analysis and development of AFTQC to heterogeneous WSNs where sensors may have vastly different processing capacities; (2) extending the analysis to source-driven WSNs where sensor readings are reported back to the PC asynchronously or periodically; and (3) extending the analysis to consider the use of multi-dimensional along different dimensions of redundancy such as "path", "source", and "modality" so that not only QoS requirements are satisfied, but also the lifetime of the system is maximized under varying network conditions. Redundancy management should utilize information available from the different network layers and should also benefit from the heterogeneous nature of WSNs. An example of modality redundancy is to use different media sensors (like acoustic and image sensors) to reinforce each others' reports whenever needed. To the best of our knowledge, solutions identifying, adaptively managing and optimizing multi-dimensional redundancy to address the tradeoff between QoS assurance and energy consumption under varying network conditions in WSNs do not yet exist. Lastly, a suggested future research direction is to build an experimental prototype to further ascertain the validity of the theory. The emphasis of the experiment would be on actual coding and implementation of the query engine, clustering, redundancy management with AFTQC and data forwarding as well as measurement for experimental validation.

APPENDIX A
ACRONYMS AND NOTATIONS

ACRONYMS[2]

MTTF	Mean time to failure, defined as the mean number of queries that the sensor system is able to answer until it fails due to channel or sensor faults, or energy depletion
SN	Sensor node
CH	Cluster head
PC	Processing center
WSN	Wireless sensor network

NOTATIONS

A	Length of each side of a square sensor area
n_b	Size of a data packet
$E_{TX\text{-}elec}$	Energy cost to run the transmitter radio circuitry per bit processed (J/bit)
$E_{TR\text{-}elec}$	Energy cost to run the receiver radio circuitry per bit processed (J/bit)
ε_{amp}	Energy used by the transmit amplifier to achieve an acceptable signal to noise ratio
E_o	Initial energy per SN
$E_{initial}$	Initial energy of the WSN
$E_{clustering}$	Energy for executing the clustering algorithm
$E_{threshold}$	Energy threshold below which the WSN depletes its energy
$E^s_{threshold}$	Energy threshold below which a SN depletes its energy
E_q	Average energy consumption per query
R_q	Probability that a query reply is delivered successfully within the deadline

[2] The plural of an acronym is spelled with an s.

r	Wireless radio communication range
p	Probability of a SN becoming a CH
q	SN failure probability
e_j	Transmission failure probability of a SN with index j
n	Number of SNs in the WSN
n_s	Number of SNs per cluster
N_c	Number of clusters in the WSN
m	Number of paths from a source CH in response to a query
m_s	Number of SNs per cluster in response to a query
f	Fraction of neighbor SNs that will forward data
λ	SN population density
d_{inter}	Distance between a source CH and the processing center
d_{intra}	Distance between a SN to its CH
N^h_{inter}	Average number of hops between a source CH and the processing center
N^h_{intra}	Average number of hops between a SN to its CH
i	Index to a path
j	Index to a node
k	Index to a neighbor node
S_{jk}	Progressive speed between two SNs with indexes j and k
$T_{clustering}$	Time interval for executing the clustering algorithm
$N_{interation}$	Number of iterations for executing the clustering algorithm
T_{req}	Query deadline requirement
R_{req}	Query reliability requirement
λ_f	Failure rate of a SN
γ	Parameter used in the calculation of the change in radio range.
θ	Parameter used in the calculation of the change in radio range. θr represents the radio circle shifted by γr
ε	Parameter used in the calculation of the change in radio range. ε is a very small number indicating the tolerance.
T_{table}	Time interval for rebuilding the best (m, m_s) table
E_{table}	Energy used for rebuilding the best (m, m_s) table

REFERENCES

[1] A. Hac, *Wireless Sensor Network Designs.* John Wiley & Sons Ltd, 2003.

[2] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, Vol. 38, No. 4, pp. 393 – 422, March 2002.

[3] D. Chen, P. Varshney, "QoS Support in Wireless Sensor Networks: A Survey," *Proceedings of the International Conference on Wireless Networks*, Las Vegas, Nevada, USA, pp. 227 – 233, June 2004.

[4] K. Romer, O. Kasten, F. Mattern, "Middleware Challenges for Wireless Sensor Networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, Vol. 6, No. 2, pp. 59 – 61, 2002.

[5] G. Khanna, S. Bagchi and Y.S. Wu, "Fault Tolerant Energy Aware Data Dissemination Protocol in Sensor Networks," *Proceedings of the International Conference on Dependable Systems and Networks*, pp. 795- 804, June 2004.

[6] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker, "Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks," Technical Report CSD-TR 02-0013, UCLA, February 2002.

[7] Z. Zhang, G. Zheng, "A Cluster Based Query Protocol for Wireless Sensor Networks," *Proceedings of* t*he 8th International Conference on Advanced Communication Technology*, Vol. 1, pp. 140 – 145, February 2006.

[8] A. Cerpa, J. Elson, M. Hamilton, J. Zhao, "Habitat monitoring: application driver for wireless communications technology," *ACM SIGCOMM Workshop on Data communication in Latin America and the Caribbean*, pp. 20-41, Costa Rica, April 2001.

[9] Y. Guo, J. McNair, **"**Fault Tolerant Three Dimensional Environment Monitoring using Wireless Sensor Networks," *Military Communications Conference*, pp. 1 – 7, Oct. 2006.

[10] B. Son, Y-S Her, K. Shim, J-G Kim, "Development of an Automatic People-Counting and Environment Monitoring System Based on Wireless Sensor Network in Real Time; To

realize in Korea Mountains," *International Conference on Multimedia and Ubiquitous Engineering*, pp. 579 – 584, April 2007.

[11] J.M. Kahn, R.H. Katz, K.S.J. Pister, "Next century challenges: mobile networking for smart dust," *Proceedings of the ACM MobiCom'99*, Seattle, Washington, USA, pp. 271–278, 1999.

[12] S. Slijepcevic, M. Potkonjak, "Power efficient organization of wireless sensor networks," *Proceedings of the IEEE International Conference on Communications ICC'01*, Vol. 2, pp. 472 – 476, Helsinki, Finland, June 2001.

[13] B. Warneke, B. Liebowitz, K.S.J. Pister, "Smart dust: communicating with a cubic-millimeter computer," *IEEE Computer*, Vol. 4, No. 1, pp. 44 – 51, January 2001.

[14] A. Chandrakasan, R. Amirtharajah, S. Cho, J. Goodman, G. Konduri, J. Kulik, W. Rabiner, A. Wang, "Design considerations for distributed micro-sensor systems," *Proceedings of the IEEE 1999 Custom Integrated Circuits Conference*, San Diego, CA, USA, pp. 279–286, May 1999.

[15] P. Bonnet, J. Gehrke, P. Seshadri, "Querying the physical world", *IEEE Personal Communications*, Vol. 7, No. 5, pp. 10–15, October 2000.

[16] T. Imielinski, S. Goel, "DataSpace: querying and monitoring deeply networked collections in physical space," *ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE 1999)*, Seattle, Washington, USA, pp. 44 – 51, 1999.

[17] S. Cho, A. Chandrakasan, "Energy-efficient protocols for low duty cycle wireless microsensor," *Proceedings of the 2001 IEEE International Conference on Acoustic, Speech and Signal Processing*, Vol. 4, pp. 2041 – 2044, 2001.

[18] W.R. Heinzelman, J. Kulik, H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," *Proceedings of the ACM MobiCom'99*, Seattle, Washington, USA, 1999, pp. 174–185.

[19] L. Yu, N. Wang, X. Meng, "Real-time forest fire detection with wireless sensor networks," *Proceedings of the 2005 International Conference on Wireless Communications, Networking and Mobile Computing*, Vol. 2, pp. 1214 – 1217, Sept. 2005

[20] James Reserve Director's Notebook, "NEON and the Terrestrial Ecology Observing Systems Laboratory", December 2006; http://www.jamesreserve.edu/news.html.

[21] Alert System Organization, "ALERT Systems," December 2007; http://www.alertsystems.org.

[22] UCLA, "Terrestrial Ecology Observing Systems," December 2007; http://research.cens.ucla.edu/areas/2007/Terrestrial/.

[23] N. Noury, T. Herve, V. Rialle, G. Virone, E. Mercier, G. Morey, A. Moro, T. Porcheron, "Monitoring Behavior in Home Using a Smart Fall Sensor and Position Sensors," *IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine and Biology*, pp. 607–610, October 2000.

[24] J.M. Rabaey, M.J. Ammer, J.L. Da Silva Jr., D. Patel, S. Roundy, "PicoRadio supports ad hoc ultra-low power wireless networking," *IEEE Computer Magazine*, pp. 42–48, 2000.

[25] N. Bulusu, D. Estrin, L. Girod, J. Heidemann, "Scalable coordination for wireless sensor networks: self-configuring localization systems," *International Symposium on Communication Theory and Applications (ISCTA 2001)*, Ambleside, UK, pp. 44-48, July 2001.

[26] M. Ogawa et al., "Fully Automated Biosignal Acquisition in Daily Routine Through One Month," *International Conference on IEEE-EMBS*, Hong Kong, pp. 1947 – 1950, 1998.

[27] Y.H. Nam et al., "Development of Remote Diagnosis System Integrating Digital Telemetry for Medicine," *International Conference on IEEE-EMBS*, Hong Kong, pp. 1170 – 1173, 1998.

[28] P. Bauer, M. Sichitiu, R. Istepanian, K. Premaratne, "The mobile patient: wireless distributed sensor networks for patient monitoring and care," *Proceedings 2000 IEEE/EMBS International Conference on Information Technology Applications in Biomedicine*, pp. 17–21, 2000.

[29] B. Sibbald, "Use computerized systems to cut adverse drug events: reports," *Canadian Medical Association Journal*, Vol. 164, No. 13, 2001; http://www.cmaj.ca/cgi/content/full/164/13/1878-a.

[30] E.M. Petriu, N.D. Georganas, D.C. Petriu, D. Makrakis, V.Z. Groza, "Sensor-based information appliances," *IEEE Instrumentation and Measurement Magazine*, pp. 31 – 35, December 2000.

[31] C. Herring, S. Kaplan, "Component-based software systems for smart environments," *IEEE Personal Communications*, pp. 60 – 61, October 2000.

[32] I.A. Essa, "Ubiquitous sensing for smart and aware environments," *IEEE Personal Communications*, pp. 47 – 49, October 2000.

[33] N. Priyantha, A. Chakraborty, H. Balakrishnan, "The cricket location-support system," *Proceedings of the ACM MobiCom'00*, pp. 32 – 43, August 2000.

[34] G.J. Pottie, W.J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, Vol. 43, No. 5, pp. 551 – 558, 2000.

[35] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks," *Proceedings of the ACM MobiCom'01*, Rome, Italy, pp. 272–286, July 2001.

[36] F. Li, G. Yu, X. Yang, C. Li, Z. Feng, "A Slack Factors Based Real-Time Query Processing Approach in Wireless Sensor Networks**,"** *Seventh International Conference on Web-Age Information Management Workshops*, pp. 3 - 10, June 2006.

[37] G. Khanna, S. Bagchi and Y.S. Wu, "Fault Tolerant Energy Aware Data Dissemination Protocol in Sensor Networks," *Proceedings of the International Conference on Dependable Systems and Networks*, pp. 795- 804, June 2004.

[38] A. Badri, F. Gu and A. Ball, "Fault-Tolerant Sensor Systems - Alternative Approaches and the Development of a Complex Practical Demonstrator," *Maintenance And Reliability Conference*, Gatlinburg, Tennessee, USA, 2001.

[39] J.Y. Chen, Y.S. Shue, H. Ogunleye and S. Bagchi, "A Comparative Study on Data Fault Tolerant Requirements for Data Propagation in Sensor Networks," Technical Report, Purdue University, June 2003.

[40] B. Krishnamachari and S.S. Iyengar, "Efficient and Fault-Tolerant Feature Extraction in Wireless Sensor," *LNCS 2634*, F. Zhao and L. Guibas (Eds.), pp. 488 – 501, 2003.

[41] B. Krishnamachari and S.S. Iyengar, "Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks," *IEEE Transactions on Computers*, Vol. 53, No. 3, pp. 241 - 250, March 2004.

[42] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentell, "Fault Tolerance Techniques for Wireless Ad Hoc Sensor Networks," *1st IEEE International Conference on Sensors*, Orlando, Florida, USA, pp. 1491 – 1496, June 2002.

[43] D. Ganesan, R. Govindan, S. Shenker and D. Estrin, "Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, Vol. 5, No. 4, pp. 11-25, 2001.

[44] C. Intanagonwiwat, R. Govindan, and D. Estrin. "Directed diffusion: A scalable and Robust Communication Paradigm for Sensor Networks," *ACM Mobicom*, Boston, MA, USA, 2000.

[45] Y. Sankarasubramaniam, O. B. Akan, and I. F. Akyildiz. "ESRT: Event-to-sink Reliable Transport in Wireless Sensor Networks," *4th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Annapolis, MD, USA, pp. 177 – 188, June 2003.

[46] D. Chen, P. Varshney, "QoS Support in Wireless Sensor Networks: A Survey," *International Conference on Wireless Networks (ICWN 2004)*, Las Vegas, Nevada, USA, June 21-24, 2004.

[47] K Sohrabi, J. Gao, V. Ailawadhi, G. Pottie, "Protocol for Self-Organization of a wireless Sensor Network," *IEEE Personal Communications*, pp. 16-27, October 2000.

[48] T. He, J. Stankovic, C. Lu, T. Abdelzaher, "SPEED: A. Stateless Protocol for Real-Time Communication in Sensor Networks," *Proceedings of 23rd International Conference on Distributed Computing Systems*, pp. 46 – 55, May 2003.

[49] B. Deb, S. Bhatnagar and B. Nath, "ReInForM: Reliable Information Fowarding using Multiple Paths in Sensor Networks," *28th Annual IEEE Conference on Local Computer Networks (LCN 2003)*, Bonn, Germany, pp. 406 – 415, October 2003.

[50] M Perilo, W. Heinzelman, "Providing Application QoS through Intelligent Sensor Management", *Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications (SNPA 03)*, pp. 93 – 101, May 2003.

[51] E. Felemban, C. G. Lee, and E. Ekici, "MMSPEED: Multipath Multi-SPEED Protocol for QoS Guarantee of Reliability and Timeliness in Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, Vol. 5, No. 6, pp. 738 – 754, June 2006.

[52] R. Iyer, L. Kleinrock, "QoS Control for Sensor Network," *IEEE International Conference on Communications (ICC 2003)*, pp. 517 – 521, May 2003.

[53] B. Deb, S. Bhatnagar, B. Nath, "Information Assurance in Sensor Networks," *Proceedings of Second International ACM workshop on Wireless Sensor Networks and Applications*, San Diego, CA, USA, pp. 160 – 168, September 2003.

[54] E. Felemban, C. Lee, E. Ekici, R. Boderm S. Vural, "Probabilistic QoS Guarantee in Reliability and Timeliness Domains in Wireless Sensor Networks," *Proceedings of IEEE INFOCOM 2005,* Vol.4, pp. 2646 – 2657, March 2005.

[55] M. Perillo, W. Heinzelman, "Optimal Sensor Management under Energy and Reliability Constraints," *IEEE Wireless Communication and Networking*, Vol. 3, pp. 1621 – 1626, March 2003.

[56] S. Bhatnagar, B. Deb, and B. Nath, "Service Differentiation in Sensor Networks," *Proceeding of the Fourth International Symposium in Wireless Personal Multimedia Communications*, Sept. 2001.

[57] H.O. Tan, and I. Korpeoglu, "Power Efficient Data Gathering and Aggregation in Wireless Sensor Networks," *SIGMOD Record*, Vol. 32, No. 4, pp. 66 – 71, December 2003.

[58] S Lindsey, C. Raghavendra, "Pegasis: Power-Efficient gathering in sensor information systems," *Proceedings of IEEE Aerospace Conference*, pp. 1125 – 1130, March, 2002.

[59] O. Younis and S. Fahmy, "HEED: A Hybrid Energy Efficient, Distributed Clustering Approach for Ad Hoc Sensor Network," *IEEE Transaction on Mobile Computing*, Vol. 3, No. 3, pp. 366 – 379, October-December 2004.

[60] W. Heinzelman, C. Chandrakasan and H. Balakrishnan, "An Application-Specific Protocol Architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communication*, Vol. 1, No. 4, pp. 660 – 670, 2002.

[61] O. Younis, S. Fahmy and P. Santi, "Robust Communication for Sensor Networks in Hostile Environments," *12th IEEE International Workshop on Quality of Service,* pp. 10 – 19, June 2004.

[62] G. Gupta, and M. Younis, "Fault Tolerant Clustering of Wireless Sensor Networks," *Wireless Communication and Networking* (*WCNC) 2003, IEEE,* Vol. 3, pp.1579 – 1584, March 2003.

[63] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy Efficient Communication Protocol for Wireless Microsensor Networks," *33rd Annual Hawaii International Conference on System Sciences*, pp. 3005 – 3014, 2000.

[64] S. Bandyopadhyay, and E. Coyle, "An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks," *Proceedings of the IEEE INFOCOM*, pp. 1713 – 1723, April 2003.

[65] Al-Karaki, J.N. Kamal, A.E., "Routing techniques in wireless sensor networks: a survey", *IEEE Wireless Communications*, Vol. 11, No. 6, pp. 6 – 28, December 2004.

[66] Karp, B. and Kung, H.T., "Greedy Perimeter Stateless Routing for Wireless Networks," *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, Boston, MA, USA, pp. 243-254, August, 2000.

[67] Y. Xu, J. Heidemann, and D. Estrin, "Geography Informed Energy Conservation for Ad-hoc Routing," *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 70–84, 2001.

[68] Y. Yu, D. Estrin, and R. Govindan, "Geographical and Energy-Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks," UCLA Computer Science Department Technical Report, UCLA-CSD TR-010023, May 2001.

[69] I. Stojmenovic and X. Lin, "GEDIR: Loop-Free Location Based Routing in Wireless Networks," *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems*, November 1999.

[70] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher, "Range-Free Localization Schemes for Large Scale Sensor Networks," *Proceedings of the Mobicom Conference*, 2003.

[71] I. Stojmenovic and X. Lin, "Location-based localized alternate, disjoint and multi-path routing algorithms for wireless networks," *Special Issue on Wireless and Mobile Ad-hoc Networking and Computing*, Vol. 63, No. 1, pp. 23 – 32, 2003.

[72] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less Low Cost Out Door Localization for Very Small Devices," Technical Report 00729, Computer Science Department, USC, April 2000.

[73] A. Savvides, C.-C. Han, and M. Srivastava, "Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors," *Proceedings of the 7th ACM MobiCom*, pp. 166–79, July 2001.

[74] L. Doherty, K.S. J. Pister, and L.E. Ghaoui, "Convex Position Estimation in Wireless Sensor Networks," *Proceedings of the Infocom Conference*, 2001.

[75] P. Gupta and P.R. Kumar, "Critical Power for Asymptotic Connectivity in Wireless Networks," Stochastic Analysis, Control, Optimizations, and Applications: A Volume in Honor of W.H. Fleming, W.M. McEneaney, G. Yin, and Q. Zhang (Eds.), 1998.

[76] P. Mhatre, C. Rosenberg, D. Kofman, R. Mazumdar and N. Shroff, "A Minimum Cost Heterogeneous Sensor Network with a Lifetime Constraint," *IEEE Transactions on Mobile Computing,* Vol. 4, No. 1, pp. 4 -15, 2005.

[77] S. Kulkarni, A. Iyer, C. Rosenberg, "An Address-Light, Integrated MAC and Routing Protocol for Wireless Sensor Networks," *IEEE Transaction on Networking*, Vol. 14, No. 14, pp. 793 – 806, August 2006.

[78] J. A. Gutierrez, E. H. Callaway, Jr., and R. L. Barrett, Jr., *Low-Rate Wireless Personal Area Networks*, IEEE Press, New York, NY, USA, 2004.

[79] H-W Tseng, S-H Yang, P-Y Chuang, H-K Wu, G-H Chen, "An Energy Consumption Analytic Model for A Wireless Sensor MAC Protocol," *IEEE 60th Vehicular Technology Conference*, Vol. 6, pp. 4533- 4537, September 2004.

[80] B. Yin, H. Shi, Y. Shang, "Analysis of Energy Consumption in Clustered Wireless Sensor Networks," *2nd International Symposium on Wireless Pervasive Computing*, February 2007.

[81] G. Bravos, A. Kanatas, "Energy Consumption and Trade-offs on Wireless Sensor Networks," *IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications*, Vol. 2, pp. 1279- 1283, September 2005.

[82] Q. Shi, "Power Management in Networked Sensor Radios - A Network Energy Model," IEEE Sensors Applications Symposium, pp. 1-5, San Diego, CA, USA, February 2007.

[83] R. Zhang, D. Yuan, Q. Liang, "Analysis of Energy Consumption and Lifetime of Wireless Sensor Networks," *2nd IEEE Conference on Industrial Electronics and Applications*, pp. 1159 – 1164, May 2007.

[84] E.J Duarte-Melo, M. Liu, "Analysis of Energy Consumption and Lifetime of Heterogeneous Wireless Sensor Networks," *IEEE Global Telecommunication Conference*, Vol. 1, pp. 21-25, November 2002.

[85] K. Leibnitz, N. Wakamiya, M. Murata, M-A. Remiche, "Analysis of Energy Consumption for a Biological Clustering Method in Sensor Networks," *3rd International Workshop on Measurement, Modeling and Performance Wireless Sensor Networks*, San Diego, CA, July 2005.

[86] M.N. Halgamuge, S.M. Guru, A. Jennings, "Energy Efficient Cluster Formation in Wireless Sensor Networks," *10th International Conference on Telecommunications*, Vol. 2, pp. 1571 – 1576, March 2003.

[87] G. Huang X. Li J. He, "Dynamic Minimal Spanning Tree Routing Protocol for Large Wireless Sensor Networks," *1st IEEE Conference on Industrial Electronics and Applications*, pp. 1 – 5, May 2006.

[88] I. Raicu, "Efficient Even Distribution of Power Consumption in Wireless Sensor Networks," *18th International Conference on Computers and Their Applications*, Honolulu, HI, March 2003.

[89] Ns-2 Wiki, "The Network Simulator/Network Emulator," October 2007; http://nsnam.isi.edu/nsnam/index.php/Main_Page.

[90] Rice University, "Protocol for Adaptive Mobile and Wireless Networking," October 2004; http://www.monarch.cs.rice.edu/.

[91] A. Park, A. Savvides, M. Srivastavas, SensorSim: "A Simulation Framework for Sensor Networks," *Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, Boston, MA, USA, pp. 104 – 111, 2000.

[92] S. Park, A. Savvides and M. B. Srivastava, "SensorSim: A simulation framework for sensor networks," October 2001; http://nesl.ee.ucla.edu/projects/sensorsim/.

[93] UC Berkeley, "Ptolemy," October 2005; http://ptolemy.eecs.berkeley.edu.

[94] P. Baldwin, S. Kohli, E. A. Lee, X. Liu, and Y. Zhao. Modeling of sensor nets in Ptolemy II. *Proceeding of the ACM/IEEE IPSN'04*, pp. 359 – 368, Berkeley, CA, USA, 2004.

[95] Institute for Software Integrated Systems, "Prowler: Probabilistic Wireless Network Simulator," January 2004; http://www.isis.vanderbilt.edu/projects/nest/prowler/index.html.

[96] Institute for Software Integrated Systems, "JProwler: Java Probabilistic Wireless Network Simulator," January 2004; http://www.isis.vanderbilt.edu/projects/nest/jprowler/index.html.

[97] Palo Alto Research Center Incorporated, DARPA NEST, "Rmase: Routing Modeling Application Simulation Environment," December 2003; http://www2.parc.com/spl/projects/ldc/nest/parc_nest_software.html.

[98] A Sobeih, W. Chen, J. Hou, L. Kung, N. Li, H. Lim, H. Tyan, H. Zhang, "J-Sim: A Simulation Environment for Wireless Sensor Networks," *Proceedings of the 38th Simulation Symposium*, pp. 175 – 187, April 2005.

[99] A Sobeih, W. Chen, J. Hou, L. Kung, N. Li, H. Lim, H. Tyan, H. Zhang, "J-Sim: A Simulation and Emulation Environment for Wireless Sensor Networks," December 2007; http://www.j-sim.org/v1.3/sensor/JSim.pdf.

[100] A Sobeih, W. Chen, J. Hou, L. Kung, N. Li, H. Lim, H. Tyan, H. Zhang, "J-Sim: A Simulation and Emulation Environment for Wireless Sensor Networks," *IEEE Infocom Poster/Demo Session*, Miami Florida, USA, March 13 – 17, 2005.

[101] A Sobeih, "A Simulation Framework for Wireless Sensor Networks in J-Sim," December 2003; http://www.j-sim.org/v1.3/sensor/sensornets_tutorial.htm.

[102] W. Chen, Y. Ge, G. He, C. Hu, H. Kim, L. Kung, N. Li, H. Lim, A. Sobeih, H. Tyan, H. Zhang, R. Zheng, "J-Sim Simulation Package," February 2004; http://www.j-sim.org/cgi-bin/j-sim_downloader.cgi.

[103] W. Chen, Y. Ge, G. He, C. Hu, H. Kim, L. Kung, N. Li, H. Lim, A. Sobeih, H. Tyan, H. Zhang, R. Zheng, "J-Sim Sensor Network Package," July 2006; http://www.j-sim.org/patch.html.

[104] W. Chen, Y. Ge, C. Hu, H. Zhang, R. Zheng, "J-Sim Wireless Extension Tutorial," February 2004; http://www.j-sim.org/v1.3/wireless/wireless_tutorial.htm.

[105] H. Tyan, "Evaluation of J-Sim," October 2003; http://www.j-sim.org/comparison.html.

[106] N. Merizzi, "Sensor Network Patch," September 2005; http://www.cas.mcmaster.ca/~merizzn/.

[107] C. Lee, "J-Sim for MMSPEED Protocol," December 2006; http://www.ece.osu.edu/~cglee/SensorNetwork/guide.htm.

[108] W. Chen, H. Zhang, H. Lim, "Greedy Perimeter Stateless Routing," May 2006; http://www.j-sim.org/contribute.html#GPSR.

[109] B. Karp, "Challenges in Geographic Routing: Sparse Networks, Obstacles, and Traffic Provisioning," *DIMACS Workshop on Pervasive Networking*, Piscataway, NJ, USA, May 2001.

[110] B. Karp, H.T. Kung, "Greedy Perimeter Stateless Routing for Wireless Networks," *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, Boston, MA, USA, pp. 243-254, August, 2000.

[111] W. Ye, J. Heidemann, D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," *Proceedings of the IEEE Infocom*, New York, NY, USA, pp. 1567-1576, Vol. 3, June, 2002.

[112] V. Tippanagoudar, I. Mahgoub, A. Badi, "Implementation of the Sensor-MAC Protocol for the JIST/SWANS Simulator," *Proceedings of the IEEE/ACS International Conference on Computer Systems and Applications*, Amman, Jordan, pp. 225-232, May 2007.

[113] W. Ye, J. Heidemann, D. Estrin, "Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks," *IEEE/ACM Transaction on Networking*, Vol. 12, No. 3, June 2004, pp. 493 – 506.

[114] W. Ye, J. Heidemann, D. Estrin, "A Flexible and Reliable Radio Communication Stack on Motes," USC Information Sciences Institute, *Technical Report ISI-TR-565*, September 2002.

[115] W. Ye, J. Heidemann, D. Estrin, "S-MAC Software," September 2005; http://www.isi.edu/ilense/software/smac/.