Fall 2006

# Fault Modeling in Wireless Sensor Networks

Ahmed A. Elmiligui
*Old Dominion University*

# FAULT MODELING IN WIRELESS SENSOR NETWORKS

By

Ahmed A. Elmiligui
B.S. Computer Engineering, December 2004, Old Dominion University

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

MASTER OF SCIENCE

COMPUTER ENGINEERING

OLD DOMINION UNIVERSITY
December 2006

Approved by:

Dr. Lee A. Belfore II (Director)

Dr. Min Song (Member)

Dr. Stephen Olariu (Member)

# ABSTRACT

FAULT MODELING IN WIRELESS SENSOR NETWORKS

Ahmed Elmiligui
Old Dominion University, 2006
Director: Dr. Lee A. Belfore II

A large amount of research has been done in the area of wireless sensor networks (WSN), but not much work has been done in modeling the fault tolerance and reliability of these networks. In this thesis, the fault tolerance of a WSN to node failures is studied and an analytical reliability model of the network is derived. A valid reliability model of a network could reveal an estimate of the network's performance before it is deployed.

A wireless sensor network was modeled as a k-out-of-n system and a generic fault tolerant framework for the network in terms of node losses was given. An analytical model was derived to model the reliability of the network due to energy loss. The network model consisted of several clusters of nodes that communicate to a central sink. The reliability of each node was modeled as a Poisson distribution whereas the failure rate for each node was defined in terms of the work load and energy consumption of the node. By studying the details of the network infrastructure, the arrival rate of workload to each node was modeled. The cluster was modeled as a parallel system of sensor nodes and the failure distribution of the cluster was derived in terms of the node's failure distributions. Another reliability model was derived using a generating function polynomial that shows the reliability of the network at any point in time.

A simulation of a wireless sensor network was performed using C++ and Matlab. The simulation results and the analytical model were compared and the results of the

analytical model closely matched the simulation results for the time duration that the network was defined. This proved the validity of the analytical model.

# DEDICATION

This thesis is dedicated to my Mother, Father & Brother and the beautiful city of Alexandria.

# ACKNOWLEDGMENT

I would like to express my sincere gratitude to my advisor Dr. Lee A. Belfore II for his advice, guidance and motivation throughout the research work.

I would also like to thank Dr. Stephen Olariu and Dr. Min Song for agreeing to be on my committee and for their valuable time and generous assistance.

I would also like to thank the Department of Electrical and Computer Engineering at Old Dominion University for providing partial support during the course of this work.

Finally, I would like to thank my family for their love and support through out my years at O.D.U.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# I. INTRODUCTION

Wireless sensor networks (WSN) have become an area of interest to all researchers due to the advances in microelectronics. A WSN consists of hundreds of sensor nodes that wirelessly communicate together to form a functional network that can collect and report data back to the user. The main advantage of a WSN is that it can be remotely deployed in an area where it self-organizes in such a way to allow the user to send sensing requests to it and receive the information back. It is expected that WSN will be involved in both critical military and commercial applications. Therefore, these sensor networks must be fault tolerant and reliable in order to be accepted by the society.

This research studies the fault tolerance and reliability of WSNs. The main motivation behind this work is the significance of WSNs and its various applications that could change our daily lives.

## *I.1. Literature Review*

In this section, a literature review of fault tolerance and modeling of wireless sensor networks is given. Little research has been conducted in the area of analyzing the fault tolerance of WSNs. Most research emphasizes the creation of fault tolerant algorithms but little effort has been dedicated to studies of the reliability of the network in the presence of faults. Moreover, no definitive summary of fault and reliability models for sensor networks exists in the scholarly literature. On the other hand, there has been work done in the field of performance modeling of WSNs.

### I.1.1. Network Modeling

In **[14]**, the authors present a method to model the performance of wireless sensor networks in terms of data capacity, data delivery, and energy consumption. The main contribution of the authors is that they use Markovian techniques to model the behavior of a single sensor and the dynamics of the entire network. The network consists of stationary nodes that report to the sink node through multi-hop communications. A key assumption is that the sensors cannot run out of energy. Each sensor contains a data buffer and can be either in a sleep state or in an active state. They break down the model into three building blocks that are the sensor node, the network model, and the network interference model as shown in Figure 1. The sensor is modeled by a discrete-time Markov chain model where time represents the data unit transmission time. The network is modeled as an open queuing network where a queue corresponds to the data buffer of a sensor. The interference model is used to compute the probability that a data unit is transmitted in a time slot for each sensor. These blocks are combined together to obtain a global system solution that does not require parameter values from an actual network simulation. The overall solution of the system is obtained by means of a fixed point approximation in which the three blocks interact by exchanging various parameters along a fixed loop until a final equilibrium is reached. These parameters include the probability state for each sensor, the transmission rates of network and the probability of failure in the interference model.

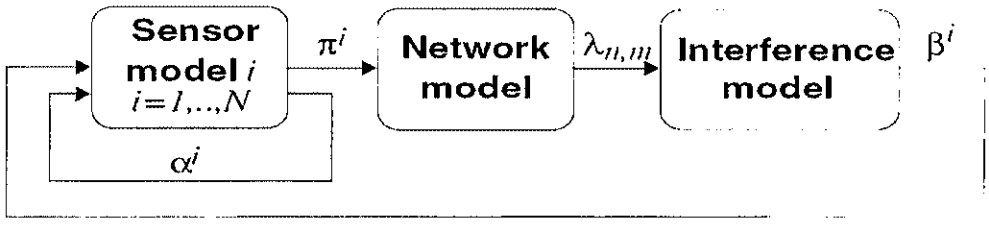



**Figure 1: Closed Loop of the network model in [14].**

In **[15]**, the authors present a modeling technique to model a sensor network in terms of adversary and network attacks. Their model consists of a cluster based hierarchical network. They use a Markov chain to model the different states that each cluster could be in from a security point of view. The cluster starts in a healthy state and gets subjected to various attacks. Once an attack occurs, the cluster can either detect and repair the attack or fail. Using this model, the steady state parameters of the Markov chain are derived through analytical work and simulations.

In **[18],** the authors model the network from a connectivity perspective by using unit disc graphs, random point processes and Bernoulli nodes. The unit disc graph models the transmission range of a node and the Bernoulli model is used for the fault tolerance of the node. Each node is either active or inactive with a Bernoulli model. A direct link exists between two neighboring nodes if they are both active. The random point process is used to model the node distribution. Using this model, the sensor network is divided into two parts that are connectivity and fault tolerance of node failures. By using the geometry and rules of the disc model, the probability of each node having at least one active node is derived. They also show the result of the connectivity model, that is the cardinality of one connected component on the connectivity graph is either one or tending to finite.

### I.1.2. Fault Tolerance Modeling

Fault tolerance has been modeled in previous literature using a variety of different approaches. All of the approaches use state diagrams to represent the different states of a network due to failures. In the first approach, fault tolerance is modeled by measuring the ability of a network to perform a certain objective in the presence of node failures. Another approach modeled fault tolerance by the ability of the network to detect gateway failures and repair them. The last approach modeled fault tolerance as a probabilistic study in terms of energy loss and work load.

In **[11,12]**, the sensor network is modeled as a static linear system and fault tolerance is defined with respect to a given estimation objective called *z*, which is a given function of the system state. This estimation objective is observable as long as there are enough sensor nodes to estimate it. A given function of the system state should remain observable when sensor failures occur. A set of sensors is pseudo-minimal (PMSS) if and only if they are the only set of sensors that can keep a functional system observable. The authors create a PMSS subgraph that is a multi-level graph where each level contains nodes with the same cardinality. The graph is then used to find the set of sensors that keeps *z* observable. The authors also map the reliability concept to time by modeling the reliability of each node as a Poisson distribution and finding the MTTF failure of the set of PMSS sensors.

In **[13]**, authors use the same concept but they provide a sensor network state automation that represents node failures. Each state represents a set of functional sensors. There are two types of states: states with enough nodes to keep *z* observable and states that do not have enough nodes to keep *z* observable. Transitions between states occur

when a node fails. Therefore, for a set of $i$ nodes there will be a total of $2^i$ states. The system is observable as long as it is in a state that has enough nodes to keep $z$ observable. Thus, the analysis of the fault tolerance of the estimation of $z$ is reduced to analyzing the paths of the sensor network state automation that culminate in an observable state.

In **[10]**, authors introduce the concept of fault tolerance by dividing the network into clusters and gateways and proposing a mechanism to recover sensors from failed clusters. The network consists of several clusters where each cluster contains only one gateway node. The gateway nodes then communicate to the command node. Therefore, a node can only communicate to the command node through the gateways. The faults that are considered are hardware and software faults that alter the operational behavior of gateways. Such faults include transmitter faults and communication faults between the gateway and the nodes that can be caused due to hardware faults, energy depletion, or environmental conditions. The fault tolerance mechanism consists of detecting gateway failures and recovering from them. Detection of the faults occurs by periodic updates through inner gateway communication where gateways exchange the status of the clusters in the system. Recovery of the failures occurs by reassociating the nodes from the faulty gateway to other clusters.

In **[1]**, the authors analyze the effect of redundancy on the time to failure of wireless sensor networks. The main contribution of this work is the probability model used for the network. They define the mean time to failure in terms of the number of queries each node can perform before failing. Furthermore, they define the number of queries based on a valid energy model of the network. Their final results show the

tradeoff between fault tolerance and energy conservation for prolonging the time of the sensor network.

Other research in fault tolerance in wireless sensor networks is shown in **[27]** where the authors study the fault tolerance of event region detection in sensor networks. Furthermore, authors in **[16,17]** design fault tolerant sensor networks by adding back up schemes and improving the process of data fusion in sensor networks.

## *I.2. Problem Statement*

Although much research has been done in the WSN field, little has been done in the study of fault tolerance of these networks. Most research has focused on designing algorithms used for fault tolerant detection and designing networks that are fault tolerant. Indeed, little effort has been devoted towards evaluating the reliability and robustness of WSN under faults and predicting how long a network would take to fail.

The main goal of this thesis is to study the reliability and fault tolerance of wireless sensor networks. A valid reliability model for a network would allow the network to be studied before it is actually deployed and put into work. Such a study would reveal the performance of the network due to node failures, the average number of nodes alive at any point in time, and the lifetime of the network. This information is useful in designing and deploying the wireless sensor network.

## *I.3. Thesis Organization*

This thesis is organized into six chapters, starting with the first chapter introducing the problem statement and a survey of earlier research in this area. Chapter II provides the preliminaries needed to aid in understanding the research problem. Chapter

III explains the reliability model used and the analytical work derived. Chapter IV explains the details of the simulation and the network. It also shows the further derivations needed to model the network. The results of the simulation and analytical work are also compared in Chapter IV. Chapter V explains the possible future work in this research. Finally, Chapter VI concludes the thesis.

# II. BACKGROUND

## II.1. Wireless sensor networks

The recent advances in wireless communications and electronics have enabled the development of low-cost and low-power multifunctional sensor nodes. These low power and high frequency systems can be implemented on relatively small chips. The result is a "*mote*" or often referred to as a "*node*" which is an autonomous, compact electronic device that can sense, process, and communicate with neighboring nodes **[20]**. A sensor network is composed of a large number of these sensor nodes densely deployed in an area. Upon deployment, the sensor positions do not need to be engineered or known. This provokes the need for them to be able to self-organize **[19]**.

The main strength of a wireless sensor network is that it can be formed using thousands of these autonomous wireless nodes. In order for the nodes to create a functional network, they must have certain characteristics [21]. First of all, a sensor node must be very energy efficient. Since the network may consist of thousand of nodes, then it is infeasible to recharge each node once it runs out of energy. Therefore, the energy of a node is considered to be the most valuable resource of a node and the primary metric for analysis. Secondly, a sensor node must be very inexpensive since they cannot be recycled and are used in large quantities. Thirdly, it must be able to communicate to its neighboring nodes using wireless communication. In most cases networks are deployed in environments that do not have an installed infrastructure for communications. Fourthly, a node must be able to communicate via multi-hop (hop by hop) routes with other nodes. This technique saves energy as energy loss is proportional to the

transmission distance. Finally, each node should be able to process local data using filtering techniques and data fusion algorithms to sense data from the environment and aggregate it transforming it into useful information.

Figure 2, shown below, shows the main components of a wireless sensor node, which are the sensing unit, processing unit, transceiver, and power unit. The sensing unit consists of the actual sensor and an analog to digital converter that is used to convert the sensed data into digital information. Most of the current hardware for sensor nodes is based on RF circuit designs. However, some nodes use Bluetooth, infrared, or optical communications that require a direct line of sight.

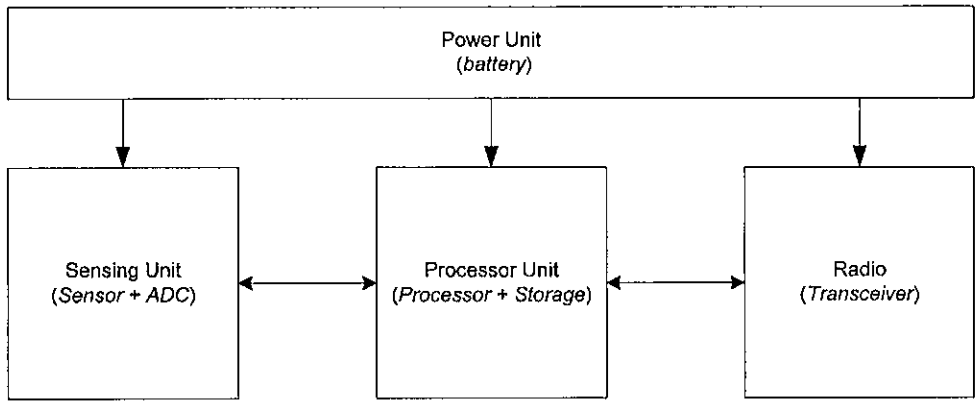



**Figure 2: An Ideal Wireless Sensor Node.**

## *II.2. Applications*

When the sensor nodes are deployed and the network is functioning properly, it can be a very useful tool for monitoring and sensing. Due to the useful capabilities of a WSN, they have been deployed in many applications in various fields such as [21,22]:

- In the military, a WSN is considered a very valuable tool where it is used for information collection, enemy tracking, battlefield surveillance and target classification. For example, sensor nodes can be deployed in a battlefield to detect

and report certain enemy movement or intrusion. Nevertheless, WSNs can also be used in peaceful military applications such as homeland security, property protection and surveillance, and border patrol.

- In environmental monitoring, where temperature and light sensors can be used in heaters, fans, and other relevant equipment to enhance there performance in a more reasonable and economic way. Also, WSNs can be used to help in detecting natural disasters such as earthquakes and tornadoes.
- In habitat monitoring, WSNs can be used for sensing temperature, measuring barometric pressure and humidity, and monitoring the wildlife and the ecosystem by tracking animal behavior.
- In agriculture fields, WSNs can be used to enhance the efficiency and growth of cultivations and monitor the level of hazardous chemicals and pesticides.
- Support for logistics by monitoring large inventories and hazardous stored chemicals. In addition, WSNs can be used to detect and classify rare events such as alarm and fault detection and periodic events such as tracking of material flows.
- In health applications, WSNs can be used to monitor the body temperature and perform health diagnosis for patients.
- In manufacturing, WSNs can aid in monitoring the quality of a products. WSNs can also be applied in unreachable places of wired networks such as bearing of a motor, oil pumps, and in unpleasant or impractical hazardous environments such as in chemical factories.

## *II.3. Data Propagation*

In order for the nodes to communicate efficiently, a data link layer must exist that controls the details of data transfer such as the multiplexing of data streams, data frame detection, and medium access and error control. This data link layer is known as the MAC protocol that forms the basic infrastructure needed by the network for communication. The main function of the MAC layer is to maintain reliable point-to-point connections within the network to allow data transfer with minimal collisions.

A valid MAC protocol is characterized by several key properties **[23].** As mentioned earlier, energy is the most critical resource in a network; therefore, the MAC protocol must be very energy efficient. Furthermore, the MAC protocol must be scalable and adaptable to node changes i.e. it must be able to handle changes in a large network size and node density.

A MAC protocol must also be able to handle radio duty cycling where the radio is off by default and wakes up periodically to participate in the network. This is handled either by asynchronous or synchronous MAC protocols. In asynchronous protocols, the transmission of packets does not rely on any time synchronization between the nodes. Instead, the packet is sent for a time that is long enough to ensure that the other end receives it. On the other hand, in synchronous protocols, the nodes maintain time synchronization so the transmission occurs at the correct time.

Once a valid MAC protocol exists in the sensor network, there must exist another protocol that allows routing of messages from the sink to the nodes and vice versa. As shown in Figure 3, nodes do not directly communicate with the user. Instead, they communicate through a sink node(s). Due to the large number of nodes and the unique

characteristics of a wireless sensor network, routing within the network can be challenging [25]. Unlike other wireless networks, sensor nodes cannot have individual IDs. As a result, IP based routing protocols cannot be used in wireless sensor networks.

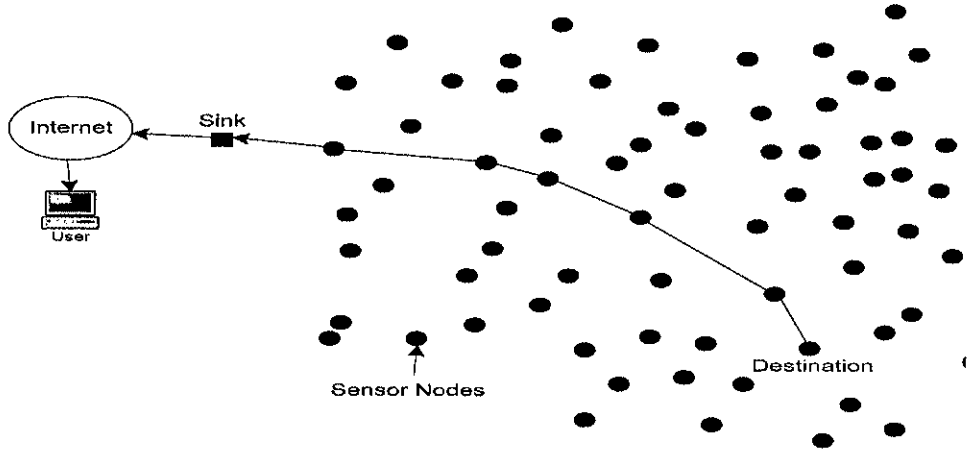



**Figure 3: Wireless Sensor Network Illustration.**

Routing protocols can be divided into three main categories: flat, hierarchical, and location based routing. In flat based routing, all nodes are assigned equal duties and responsibilities. While in hierarchical routing, nodes are grouped into clusters so the cluster head can perform aggregation and reduction of data. The higher energy nodes are used to send and receive information while the lower energy nodes are used for sensing tasks. Finally, in location based routing, position information is used to send data to direct regions rather than the whole network.

Another class of routing protocols is proactive, reactive and hybrid routing which depends on how a source finds a route to the destination. In proactive routing the routes are created before they are needed such as in table-driven routing. In reactive routes the routes are determined on demand i.e. only when a message needs to be sent. Hybrid routing uses a mixture of both.

## II.4. A Wireless Sensor Network Model

In [2] the authors provide a virtual infrastructure for WSNs by using variable signal strengths and directional broadcasts from the sink node to create node clusters. The uniqueness of this network is that it provides a coordinate system for the network using a training protocol. Once the coordinate system is setup then multi-hop communication routes are formed from the sink to all clusters in the network.

The network model consists of a single sink node and a massive number of sensor nodes that are deployed within the transmission range of the sink node. The main characteristics of the sensors are:

- Sensors are initially in sleep mode and wake up at random times for short intervals
- The sink node has a transmission range that covers the entire sensor network
- Each sensor node (other than the sink) has a limited transmission range and as a result only a limited number of sensor nodes are considered to be within a one hop range from the sink
- None of the sensors have knowledge of the topology of the network

As shown in Figure 4(a), the sink node is considered to be at the center of the coordinate system. The coordinate system consists of two dimensions $(\theta, y)$ where $\theta$ is the angle between the sink and node and $y$ is the number of hops between the sink and node. All nodes with the same angle and distance $(\theta, y)$ from the sink are considered to be within the same cluster.

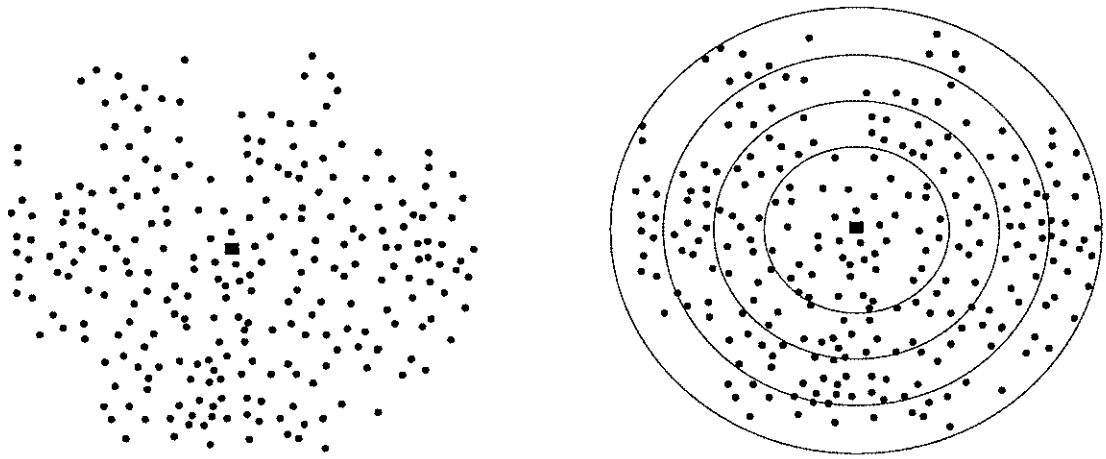
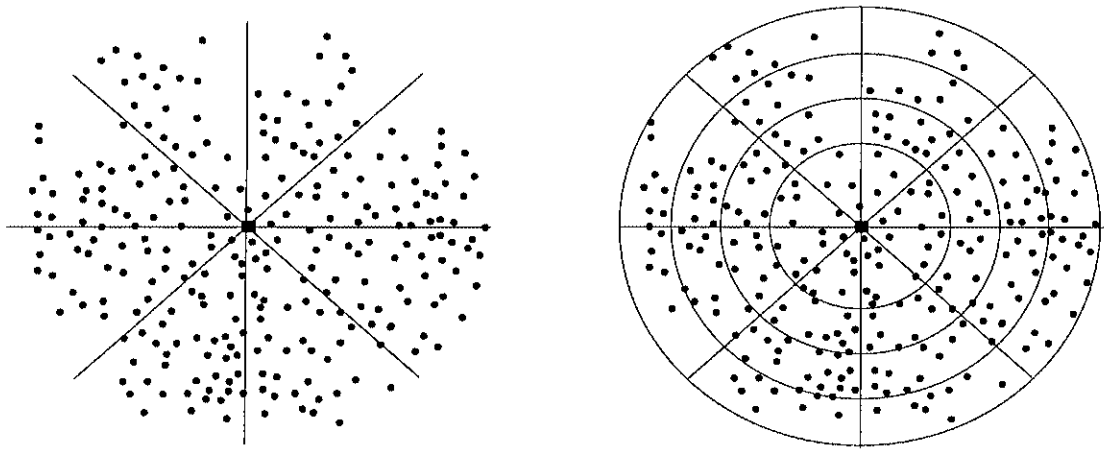



**Figure 4: (a) A Sensor Network Model with a central sink node; (b) Network divided into Wedges.**

When the network is first deployed it has no virtual infrastructure and therefore it needs to be trained so that each node will acquire its coordinates. In order to train the sensor network, the sink starts by transmitting a beacon with the lowest possible level. All nodes within the radius of this beacon that receive it with a certain quality will now belong to the first corona. The beacon is then sent again at a higher power level to capture nodes of the second corona. This process is repeated until all nodes within the sensor are assigned to a corona as shown in Figure 4(b).



(c) (d)

**Figure 5: (c) Network divided into Coronas; (d) A trained sensor network.**

Similarly, in Figure 5(c), nodes of different wedges are differentiated using the same concept. The sink will broadcast a beacon using a reduced angle aperture to a small wedge of the sensors. All nodes that receive this signal with a certain quality are now members of the first wedge. This process is repeated until all nodes are assigned to a wedge.

At this point, all nodes now are assigned with the two coordinates ($\theta$, $y$). Nodes with the same corona number and wedge number are grouped into one cluster. Figure 5(d) shows the trained network with the overlying coordinate system. This network has four coronas and eight wedges which results in thirty-two clusters. Each cluster connects to a cluster head that is responsible for communicating to neighboring clusters and collecting data from cluster nodes.

Routing in the network occurs by multi-hop transmission through the clusters. Each cluster head collects the information from the cluster nodes and transmits it to the neighboring cluster within the same wedge. As a result, each cluster sends to only one

neighboring cluster and therefore eliminating MAC level contention in the routing process.

## *II.5. Fault Tolerance*

### II.5.1. Definition

Fault tolerance can be defined as the ability of a functional unit to perform a required task in the presence of faults and errors **[4]**. In the fault tolerant computing literature, performability is a measure that can be used to evaluate the system's degradation over time. In other words, a fault tolerance study of a system reveals how many faults a system can handle before it fails. Such a study also exposes the degradation of the system quality and functionality over time. One of the main parameters used to measure fault tolerance is the reliability of a system which can be defined as the ability of a unit to perform a required task in the presence of faults and errors for a stated period of time.

The reliability of a system is defined as the probability that a system operates, without fail, over a desired time interval. The four basic elements that define reliability are probability, adequate performance, time and operating conditions **[26]**. Probability is one of the most important parameters of reliability as it specifies the chances of a failure occurring at a given time. This probability is usually obtained either from previous performance of a system or valid mathematical models. Adequate performance specifies the conditions in which a unit or system is considered to be failed. For example, a computer node on a network might be considered as failed if it is physically damaged or it is no longer connected to the network. Therefore, the failure modes must be first known in order to specify the adequate performance of a system. Time specifies the failure rate

of the system, that is the number of failures per unit time. The failure rate of a component is usually closely related to the operating conditions in which it functions.

### II.5.2. Why Study Fault Tolerance?

Engineers have always been trying to design systems that are free from any form of flaw or error; however, in reality, this can never occur. Engineering systems can be used in many different critical applications such as in the military and industry. If the failures of these systems are not properly modeled, the design of these systems can be inadequate, resulting in major accidents and inconveniences, because the system's behavior in the presence of faults was not adequately considered in the design. If the faults in a system can be accurately modeled before a system is deployed, then this can reduce the cost of the system, prolong its lifetime and reduce accidents and malfunctions. For example, consider a factory that has four parallel manufacturing lines. If the number of average units that each line can produce before failing is known, then this machine line could be replaced before it fails rather than waiting for it to fail and then having to repair it.

### II.5.3. Fault Models

Assuming that the faults that occur in a unit are independently and identically distributed, then the reliability distributions can be defined using the Poisson distribution

$$R(t) = e^{-\lambda * t} \tag{1}$$

$$F(t) = 1 - e^{-\lambda * t} \tag{2}$$

where $\lambda$ is the number of failures per unit time, R(t) is the reliability probability, and F(t) is the unreliability probability.

For systems that consist of several units connected in parallel where there are redundant units to increase the reliability of the system and all units are stochastically independent, then the lifetime distribution of the entire system is modeled as a parallel system

$$F_S(t) = \prod_{i=1}^{m} 1 - F(t) = \prod_{i=1}^{m} 1 - e^{-\lambda_i * t} \tag{3}$$

where $F_S(t)$ is the lifetime distribution of the system, $m$ is the number of units in the system, $F(t)$ is the lifetime distribution of an individual component and $\lambda_i$ is the failure rate of unit number $i$.

A unique model exists in the case where there are redundant units in a system [7]. Assuming a system of $n$ components which works if and only if $k$ components work, then this system is called a k-out-of-n system and can be modeled using the following polynomial

$$g_n(z) = \prod_{i=1}^{n} (q_i + p_i z) = \sum_{i=0}^{n} R_e(i,n) z^i$$

where $q_i$ is the unreliability of component $i$, $z$ is a dummy variable, $p_i$ is the reliability of component $i$ and $R_e(i,n)$ is the probability that exactly $i$-out-of-$n$ components are working.

# III.THEORY

The ultimate goal of this research is to model the energy failure of wireless sensor networks. In order to accomplish this, the failure modes of the network must be first identified. Furthermore, the tolerance of the network to these failure modes must be studied. This chapter creates a generic fault model for wireless sensor networks and defines the different failure rates in the network. In addition, it applies these theories on a network model.

## *III.1. A Generic Fault Model for Wireless Sensor Networks*

As stated earlier, a wireless sensor network can be exposed to a various number of faults. Although the faults may arise from many different causes, they all have the same effect which is a permanent loss in one or more of the sensor nodes. When a network is first deployed in an area, it has all of its nodes alive. Throughout the lifetime of the network, different nodes will start to fail due to energy depletion. The network will eventually reach a point where it has lost the maximum number of nodes that can be tolerated to achieve its desired function. Assuming the number of nodes required for the network to operate is $k$ nodes and the initial number of nodes at deployment is $n$ nodes, then a sensor network could be modeled as a $k$-out-of-$n$:G system structure. This means that the system is considered to be functional if and only if at least $k$ out of the $n$ components (nodes) are alive.

A $k$-out-of-$n$:G network could be modeled using $n-k$ states where a state transition occurs for every node failure. Each state of the network is defined as the current number of nodes alive in the network. A node failure will directly change the

state of the network as the network becomes one more step closer to failing. Independent of the degree of node redundancies and failure rates, a node failure will always cause a permanent loss of a network resource.

Some middleware solutions are able to change the properties of the network and modify their own behavior according to the network conditions [9]. Therefore, adjusting the amount and quality of the job requests to the network with respect to node failures can increase the overall network lifetime. For example, if the network is experiencing many node failures and there are only a limited number of nodes left, then the middleware can lower the QoS and processing rate of the workload. However, this does not need to be done for every node failure but rather only at certain failure milestones. There are two main milestones in the lifetime of a network when considering node failures. The first one is when the initial node failure occurs. After the first failure, the network will continue to operate normally until it reaches a critical point where there are no more node redundancies and any additional failures will result in a network failure. This is considered to be the second milestone. Figure 6 shows a state diagram of the network.

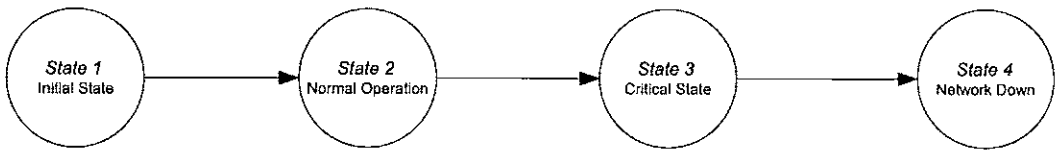



**Figure 6: State Representation of Node Failures in a *k*-out-of-*n* Network.**

When the network is first deployed, it is in State 1 and it has all of its $n$ nodes working correctly ($\chi$ nodes). When the first node fails, the network will enter State 2, which represents the *normal operation* of a network. The network will remain in State 2 until it experiences $\chi$ node failures where $\chi = n - k$. Once $\chi$ node failures have occurred

the network will be in state 3, which represents the *critical state* and any more failures will cause the network to fail. Once the network fails, then it is in state 4 (*network down*). This state does not specifically mean that all nodes are failed; however, enough nodes have failed so that the network can no longer perform its required tasks. Table 1 shows a summary of the number of nodes alive and failed in each state and the status of the network.

**Table 1: Description of Node Failure States**

| | Number of Nodes Alive | Number of Failed Nodes | Status of the Network |
|---|---|---|---|
| **State 1** | $n$ nodes alive | zero | Just deployed |
| **State 2** | $n$-$\chi$ nodes alive; where $n$-$\chi$ > k | $\chi$ where $k < \chi < n$ | Normal Operation. Tolerable faults. |
| **State 3** | $k$ | $n$-$k$ | Last state before failure |
| **State 4** | Less than $k$ nodes | More than $k$ nodes | Failed |

## *III.2. Defining the Network Failure Rate*

Even though Figure 6 shows the main states of the network, it does not represent when the network enters these states i.e. time. As explained in **[4]** the main parameter of interest in reliability engineering is the failure rates of the components. The failure rate is defined as the number of components that fail per unit time that, in a sensor network, is considered the number of node failures per unit time. In order to study the failure rate of the whole network, then the failure rate of each component must be studied first. Furthermore, the factors that cause the node failures must be identified.

### III.2.1. Reliability Distribution

In reliability engineering, the exponential distribution is often used to model failures in a system. While the exponential distribution does not provide an exact model for component failures, the distribution has features that resemble the actual failure behavior while providing a powerful analytical framework. Assuming a constant failure rate for each node of $\lambda_n$ then the reliability distribution (survival probability) and the un-reliability distribution (failure probability) of a node is **[3]**:

$$\text{R(t)} = \text{Reliability of node: } e^{-(\lambda t)} \tag{4}$$

$$\text{F(t)} = \text{Unreliability of node : } 1 - e^{-(\lambda t)} \tag{5}$$

Figure 7 shown below shows the unreliability of a node using equation (5) plotted for different failure rates of 0.2, 0.4, 0.6, and 0.8 failures per time unit. As shown, for a smaller failure rate, the exponential rate of the probability failure decrease; this results in a longer lifetime of the network. This graph agrees with the result in [6] where the authors deployed a wireless sensor network for four months and showed that the probability of a node failure follows this same exponential failure.

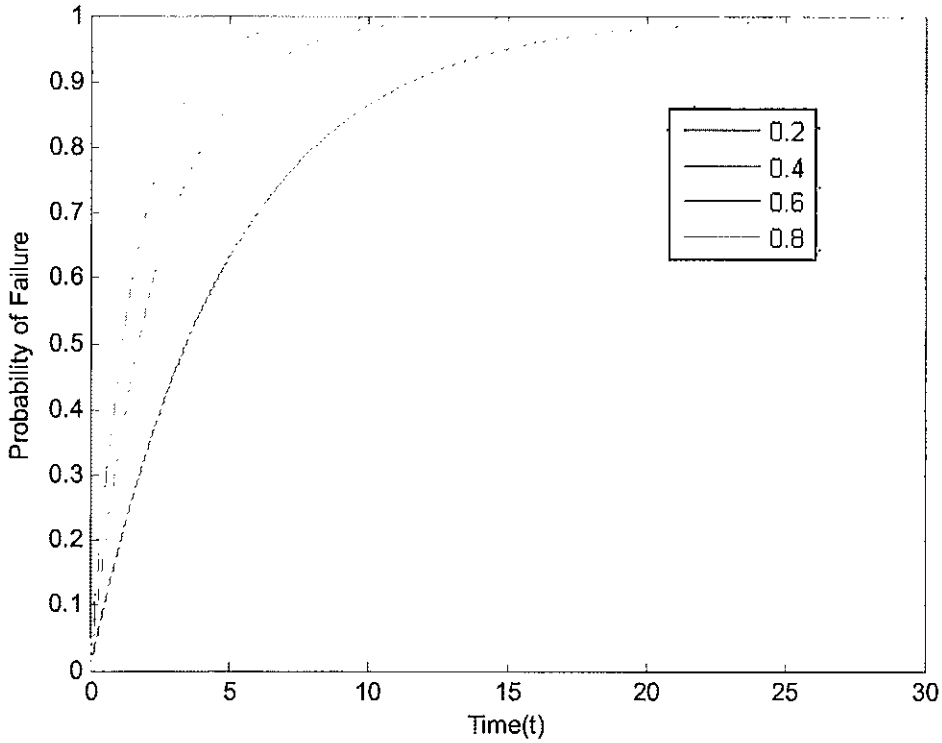



**Figure 7: Plot of Failure Probability vs. Time with Different Failure Rates.**

## III.2.2. Node Failures

The two main factors that cause a node to fail are energy and physical failures. A physical failure occurs when a node is damaged or lost due to harsh environmental factors. For example, a sensor network that is deployed in a military field might experience many physical node failures due to military machinery damaging them or external disturbances. For example, a network that is deployed in severe weather might experience many failures due to strong winds that blow the node out of the connectivity area.

### III.2.3. Energy Failure Rate:

The main event that causes a node to consume a certain percentage of its energy reserve is when it receives a job request from the sink node or the cluster head. A job request is defined as any request that the node receives that causes it to lose energy such as forwarding messages, broadcasting, sensing and sending data. Therefore, the rate at which the energy is consumed depends on the arrival rate of jobs to the node and the energy consumption of each job. Since there is always a limited energy level in the node, a node can only do a finite number of jobs [1]. Let $N_J$ denote the average number of jobs a node can perform. In order to determine the value of $N_J$, the fraction of node energy each job consumes must be known. Let $\alpha_E$ denote the fraction of energy lost for every job that the node receives and $E_N$ denote the amount of energy in joules a node depletes before failing then:

$$N_J = \frac{Job}{\alpha_E joules} * \frac{E_N\ joules}{failure} \tag{6}$$

The rate of nodes lost due to energy failures can be then defined as:

$$\frac{\text{Number of Failures}}{Time} = \frac{\lambda_J \text{ Jobs}}{Time} * \frac{\text{Failure}}{N_J \text{ Jobs}} \tag{7}$$

where $\lambda_n$ is the arrival rate of the jobs to the node

Equation (7) could be used to calculate the failure rate at any level in the network such as at the node, cluster, or whole network level. This is due to the fact that the units of $\lambda_J$ and $N_J$ cancel out and the result is in failures per time unit. However, both $\lambda_J$ and $N_J$ must be calculated at the correct network level. For example, to calculate the failure

rate at a node level then $\lambda_J$ must be calculated in terms of job arrivals to the node per unit time and $N_J$ must be calculated in terms of number of jobs a node can handle before failing.

A node can serve multiple types of jobs where each job consumes a different energy level i.e. each job has a different $\alpha_E$ value. For example a job to forward a message to another node will usually require less energy than that of a job to sense an environmental variable and send the result. For the case of more than one job type where each job type is independent, the final failure rate will be the sum of the failure rate for each job type calculated using equation (7). Assuming there are *i* different jobs then the failure rate can be computed as:

$$\frac{\text{Number of Failures}}{Time} = \left(\frac{\lambda_{J1}\ \text{Jobs}}{Time} * \frac{\text{Failure}}{N_{J1}\ \text{Jobs}}\right) + \ldots + \left(\frac{\lambda_{Ji}\ \text{Jobs}}{Time} * \frac{\text{Failure}}{N_{Ji}\ \text{Jobs}}\right) \tag{8}$$

### III.2.4. Physical Failure Rate

Relatively speaking, the probability of a physical failure is expected to be much lower than the probability of energy failure. This is due to the fact that the frequency of job requests to the network is usually higher compared with the influence of harsh environment factors. In previous simulations, the probability of physical failure used varied from $10^{-8}$ to $10^{-6}$ to $10^{-3}$ depending on the environment in which the network is deployed [1].

### III.2.5. Network Failure Rate

Once the failure rate of each component is defined then the total reliability of the entire network could then be modeled as [3]

$$\prod_{i}^{m}(1-e^{-\lambda_i t}) \tag{9}$$

where $m$ is the number of nodes in the network.

Figure 8 shown below is the plot of equation (9) for different node failure rates for the nodes. As shown there are two main turning points in the probability of failure where the probability increases exponentially from zero then it increases almost linearly until it reaches a point where it increases asymptotically approaching one.

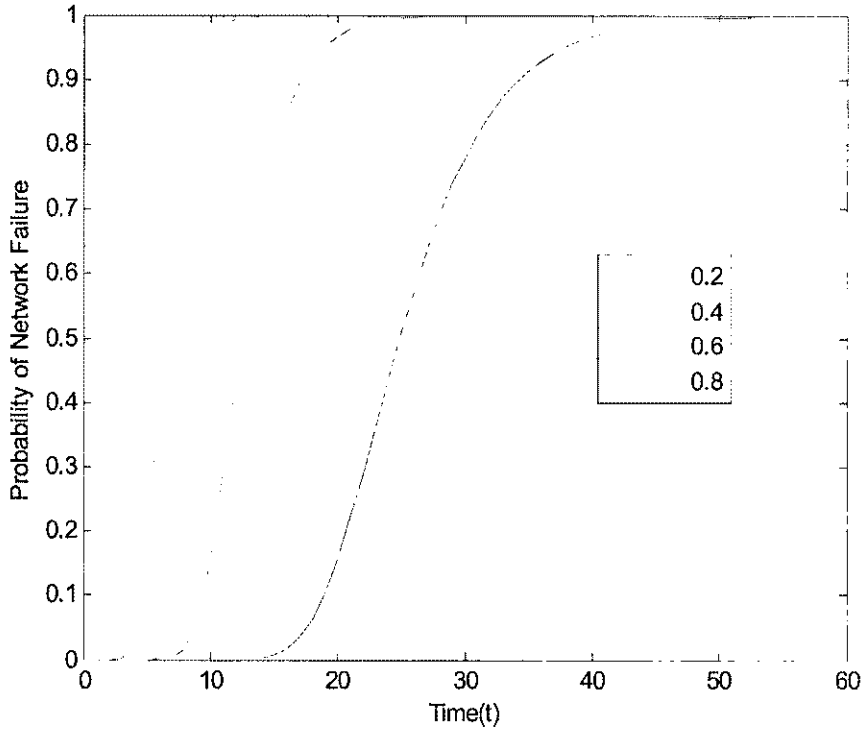



**Figure 8: Plot of Network Lifetime distribution with different failure rates.**

Equation 9 assumes that all nodes have the same failure rates. Indeed, in most wireless sensor networks, the failure rates differ among nodes because the work load is different in different parts of the network. Since the failure rate is directly proportional to

the work load on the network, most probably nodes within the same cluster will have a similar failure rate.

## *III.3. Network Model*

The network model used in this section is shown in Figure 9. The sink node, which is located in the center of the network is responsible for sending tasks to the cluster nodes. A task is defined as an action (such as sensing an environmental variable) to be performed by a chosen cluster. Each task has a QoS level and a targeted cluster ID associated with it. The QoS level specifies how many nodes of the cluster to be included in the sensing request.

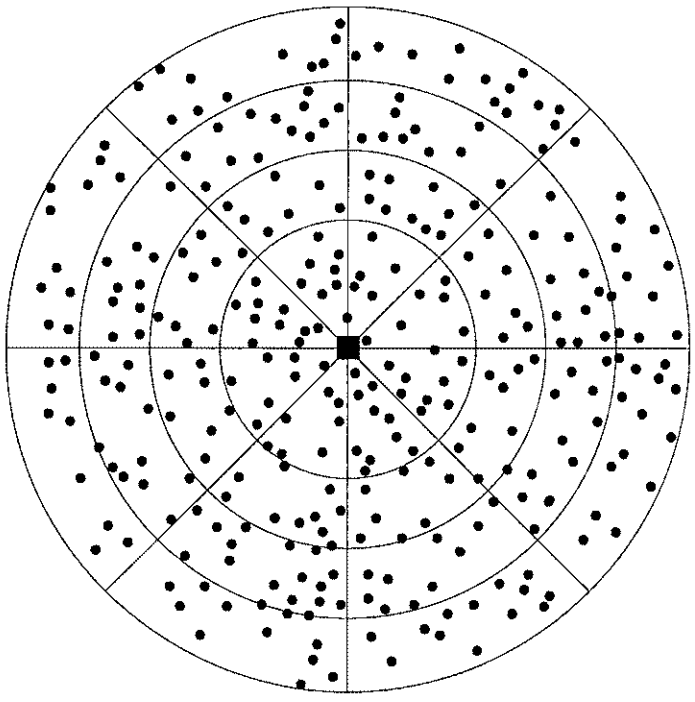


**Figure 9: Network Model.**

Tasks arrive at the sink node with a rate of $\lambda_N$. They are sent to the cluster via multi-hop transmissions within a wedge. Each cluster contains a cluster head that is

responsible for communicating with the neighboring clusters. Therefore, no MAC layer contention occurs when sending messages between clusters. A new cluster head is randomly chosen from the cluster nodes at a certain time interval. Also, the cluster head is responsible for broadcasting the task to the cluster nodes and collecting the sensed information. Communication between the cluster head and the nodes is accomplished using a Time Division Multiple Access (TDMA) technique. The sensed data is also sent back to the sink via multi-hop transmission. It is assumed that all nodes are uniformly distributed in such a way that all clusters have the same number of nodes. Furthermore, the probability of a task targeting a cluster is uniformly distributed.

Clusters of the first corona level experience the highest task arrival rates since they are located one hop away from the sink. As a result they will have the highest failure rates and will fail before the other corona levels. Once the first corona level fails, then the whole network fails because without this level the multi-hop transmission can not occur. As a result this model is only defined on the interval

$$0 \langle t \langle T_1 \tag{10}$$

where $T_1$ is the time at which all the first level clusters fail

### III.3.1. Sensor's Energy Model

In order to determine when the energy of a node is depleted, the average energy loss in a node must be determined. A sensor node mainly consists of a sensing circuit, a radio model, and a digital signal processor. The main energy parameters used to model a sensor is the energy (J) per bit dissipated in transmitter and receiver electronics, which are $E_{tx}$ and $E_{rx}$ respectively. The formula for calculating $E_{tx}$ and $E_{rx}$ is given below **[6]**

$$E_{tx} = (\alpha_t + \alpha_{amp} * d^2) * r \quad (11)$$

$$E_{rx} = \alpha_r * r \quad (12)$$

**Table 2: Parameters used in calculating $E_{tx}$ and $E_{rx}$**

| Symbol | Meaning |
|---|---|
| $\alpha_r$ , $\alpha_t$ | Energy lost in the transmitter and receiver electronics per bit |
| $\alpha_{amp}$ | Energy dissipated in transmitter amplifier |
| $r$ | Number of bits in the message |
| $d$ | Distance that the message traverse |

## III.3.2. Cluster Failure Rates

In order to model the energy dissipated in this network, the different failure rates within the network must be first defined. Since all tasks initiate from the sink node and traverse in the network through multi-hops, the clusters of the first corona will have relatively more traffic than the other coronas and have a higher failure rate. Each cluster can do one out of two jobs: it can either 1) forward a message (referred to as forwards) or 2) collect sensed data and send it towards the sink (referred to as tasks). Obviously, a task will require more than one node and therefore will consume more energy from the cluster. The failure rate of each cluster could be defined by the following equation:

$$\lambda_{cluster} = \lambda_{forwards} + \lambda_{tasks} \quad (13)$$

$$\lambda_{cluster} = \left( \frac{\lambda_F \text{ Jobs}}{Time} * \frac{\text{Failure}}{N_F \text{ Jobs}} \right) + \left( \frac{\lambda_D \text{ Jobs}}{Time} * \frac{\text{Failure}}{N_D \text{ Jobs}} \right) \quad (14)$$

where $\lambda_{cluster}$ is the failure rate of a cluster, $\lambda_{forwards}$ is the failure rate due to forwarding tasks, and $\lambda_{tasks}$ is the failure rate due to the actual sensing tasks, $\lambda_F$ is the arrival rate of forwarded messages to a cluster, $\lambda_D$ is the arrival rate of data messages to the cluster,

$N_F$ is the average number of jobs a node can do when only considering the forwarded messages, and $N_D$ is the average number of jobs a node can do when only considering the task messages.

In order to calculate $\lambda_{cluster}$, the arrival rates $\lambda_F$ and $\lambda_D$ must be first defined. Since all of the tasks are initially sent from the sink node to the outer clusters, then the arrival rate of tasks and forwards is not the same across all coronas of the network. Each corona level will then have a unique arrival rate for tasks and forwards and, therefore, a different failure rate. The failure rate of each cluster could be defined as:

$$\lambda_{cluster1} = \left( \frac{\lambda_{F1}\ \text{Jobs}}{Time} * \frac{\text{Failure}}{N_{F1}\ \text{Jobs}} \right) + \left( \frac{\lambda_{D1}\ \text{Jobs}}{Time} * \frac{\text{Failure}}{N_{D1}\ \text{Jobs}} \right) \tag{15}$$

$$\lambda_{cluster2} = \left( \frac{\lambda_{F2}\ \text{Jobs}}{Time} * \frac{\text{Failure}}{N_{F2}\ \text{Jobs}} \right) + \left( \frac{\lambda_{D2}\ \text{Jobs}}{Time} * \frac{\text{Failure}}{N_{D2}\ \text{Jobs}} \right) \tag{16}$$

$$\lambda_{cluster3} = \left( \frac{\lambda_{F3}\ \text{Jobs}}{Time} * \frac{\text{Failure}}{N_{F3}\ \text{Jobs}} \right) + \left( \frac{\lambda_{D3}\ \text{Jobs}}{Time} * \frac{\text{Failure}}{N_{D3}\ \text{Jobs}} \right) \tag{17}$$

$$\lambda_{cluster4} = \left( \frac{\lambda_{F4}\ \text{Jobs}}{Time} * \frac{\text{Failure}}{N_{F4}\ \text{Jobs}} \right) + \left( \frac{\lambda_{D4}\ \text{Jobs}}{Time} * \frac{\text{Failure}}{N_{D4}\ \text{Jobs}} \right) \tag{18}$$

where $\lambda_{cluster1}$ denotes the cluster in the first corona, $\lambda_{cluster2}$ denotes the cluster in the second corona, $\lambda_{cluster3}$ denotes the cluster in the third corona, and $\lambda_{cluster4}$ denotes the cluster in the fourth corona.

Since all nodes within a cluster are working in parallel, the cumulative failure distribution for each cluster is:

$$F_1(t) = (1 - e^{-\lambda_{cluster1} t})^n \tag{19}$$

$$F_2(t) = (1 - e^{-\lambda_{cluster2} t})^n \tag{20}$$

$$F_3(t) = (1 - e^{-\lambda_{cluster3} t})^n \tag{21}$$

$$F_4(t) = (1 - e^{-\lambda_{cluster4} t})^n \tag{22}$$

where $n$ is the number of nodes in each cluster, $F_1(t)$ is the failure distribution for a node of the first corona, $F_2(t)$ is the failure distribution for a node of the second corona, $F_3(t)$ is the failure distribution for a node of the third corona, and $F_4(t)$ is the failure distribution for a node of the fourth corona.

The average failure distribution for a wedge of the network will then be the average of the cluster failure distributions as shown

$$\lambda_{Avg} = \frac{\lambda_{cluster1} + \lambda_{cluster2} + \lambda_{cluster3} + \lambda_{cluster4}}{4} \tag{23}$$

where $\lambda_{Avg}$ is the average failure distribution of the wedge

In order to find the different arrival rates of tasks and forwards to each cluster, a detailed model of each cluster is needed. Since all of the wedges and coronas are randomly chosen with a uniform distribution, then the rates of each cluster level are identical in all of the wedges. Figure 10 shows the different rates that are present in the four different clusters of a wedge.

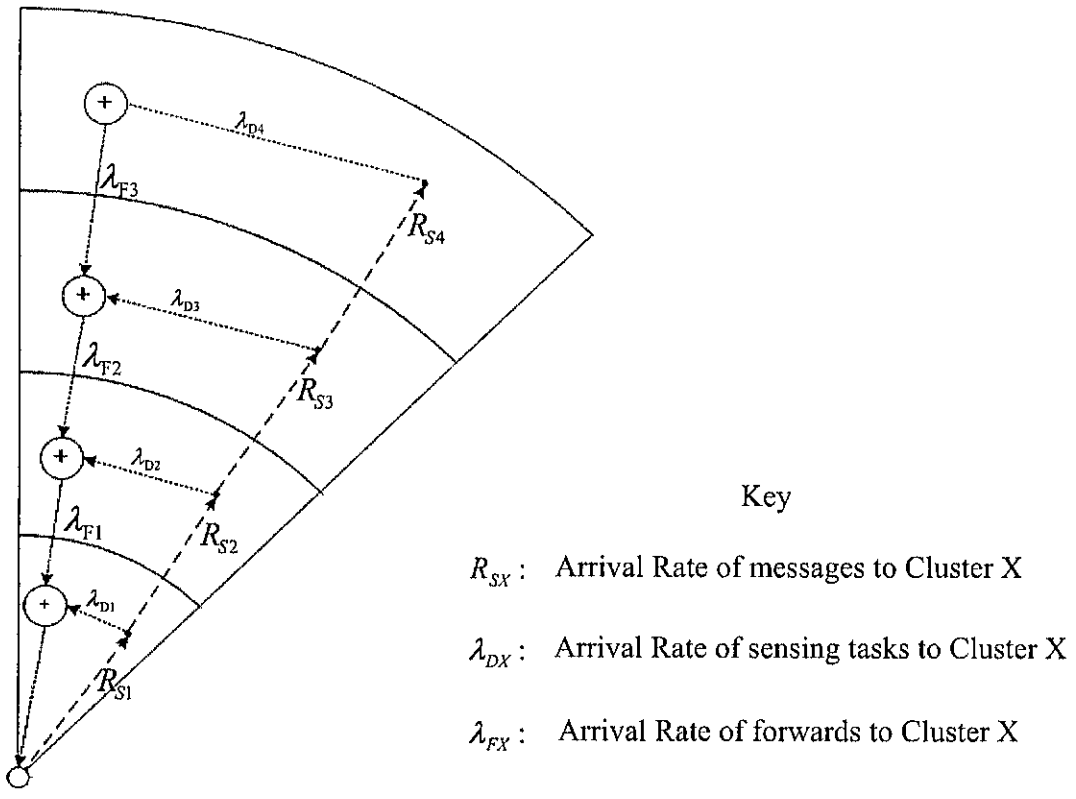



**Figure 10: Model of the different failures in the clusters.**

There are two types of messages that can arrive to a cluster head. The first type is a forward that will arrive with a certain rate from its upper cluster and is headed towards the sink. This forward will be forwarded to the lower level cluster. The second type is a task message that could either be a task for this cluster or a forward targeted to an upper level cluster. If it is a task, the data will be collected and it will be forwarded to the lower level cluster. Therefore, each cluster has two arrival rates and two forward rates. All the rates within the wedge will be a fraction of the initial arrival rate of tasks to the sink which is $\lambda_N$. As shown in Figure 10, the total arrival rate of forwards to the cluster can be defined as the sum of the forwarding rate of the upper level cluster, that is the rate of the data sent from the sensing task and forwarded data it received from its upper level cluster. This is shown in the equations below:

$$\lambda_{F1} = \lambda_{D2} + \lambda_{F2} \tag{24}$$

$$\lambda_{F2} = \lambda_{D3} + \lambda_{F3} \tag{25}$$

$$\lambda_{F3} = \lambda_{D4} \tag{26}$$

## III.4. Arrival Rates

The arrival rate of messages could be defined as the proportion of the tasks that get forwarded from the lower cluster. When a cluster receives a message from a lower level cluster, this message has only one of two possibilities. The first possibility is that it is a task for the current cluster and it will not be forwarded. The second possibility is that it is a task to one of the upper clusters and needs to be forwarded. The probability that a message is a task for the current cluster or an upper level cluster depends on the corona level. The details of each cluster are explained below.

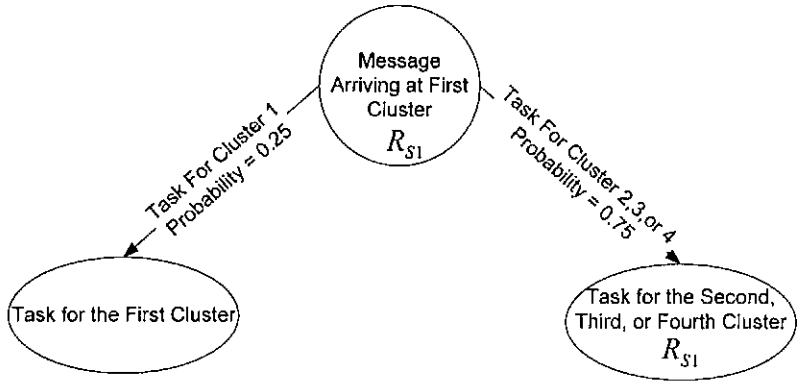



**Figure 11: Rates of the first cluster.**

Since there are eight clusters in the first corona and all coronas have the same probability of receiving a task from the sink then the arrival rate of messages to a cluster from the first corona is $\lambda_N/8$. As shown in Figure 11, the probability that an arriving

message is a task for the first cluster is *0.25*. The probability that a message is a task for one of the other three clusters is *0.75*.

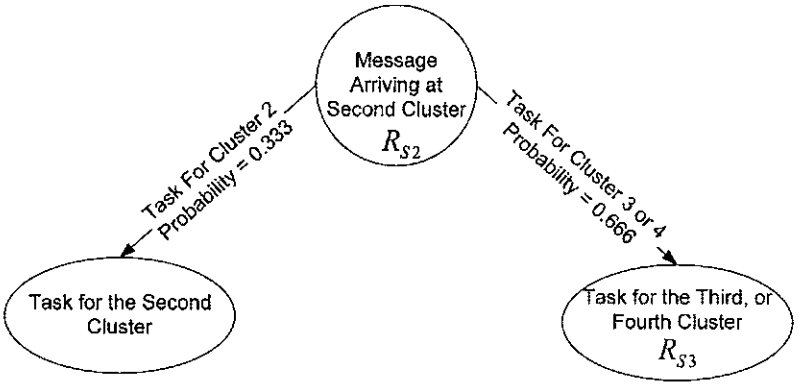



**Figure 12: Rates of the second cluster.**

For the second cluster, messages arrive at a rate of $0.75 * R_{S1}$ from the first cluster. This message is either a task for the second, third, or fourth cluster. As shown in Figure 12, the probability that this message is a task for the second cluster is 0.333. The probability that this message is a task for one of the other two clusters is 0.666.

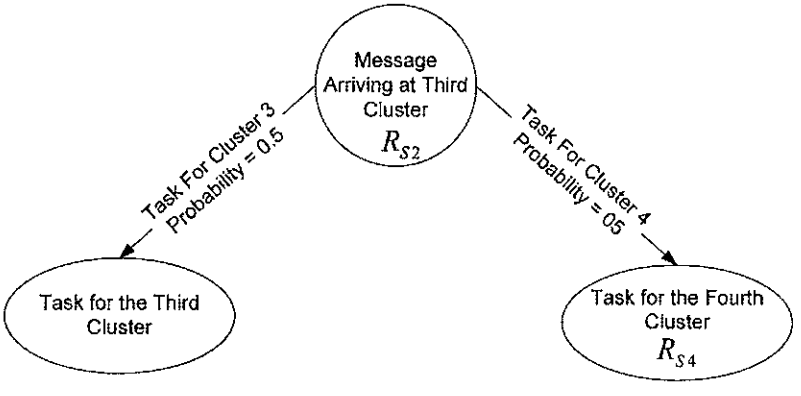



**Figure 13: Rates of the Third cluster.**

For the third cluster, messages arrive at a rate of $0.666 * R_{S2}$ from the second cluster. This message is either a task for the third or fourth cluster. As shown in Figure

14, the probability that this message is a task for the third cluster is 0.5. The probability that this message is a task for the fourth cluster is also 0.5.

As for the fourth cluster, all messages that are received are considered task messages since there are no more upper level clusters to forward to. Therefore, the probability that an arrived message is a task is one. There are no forwarded messages in the fourth cluster.

The arrival rate of messages for each cluster is summarized below in equations (27) through (30). The arrival rate of tasks to a cluster will be the proportion of the arriving messages that are tasks for this cluster. Equations (31) through (34) show the formulas for the arrival rate of tasks to each cluster.

$$R_{S1} = \frac{\lambda_N}{8} \quad (27) \qquad R_{S2} = \frac{3}{4} * (R_{S1}) = \frac{3}{32} * (\lambda_N) \quad (28)$$

$$R_{S3} = \frac{2}{3} * (R_{S2}) = \frac{\lambda_N}{16} \quad (29) \qquad R_{S4} = \frac{1}{2} * (R_{S3}) = \frac{\lambda_N}{32} \quad (30)$$

$$\lambda_{D1} = \frac{1}{4} * R_{S1} \Rightarrow \frac{1}{32} * \lambda_N \quad (31) \qquad \lambda_{D2} = \frac{1}{3} * R_{S2} \Rightarrow \frac{1}{32} * \lambda_N \quad (32)$$

$$\lambda_{D3} = \frac{1}{2} * R_{S3} \Rightarrow \frac{1}{32} * \lambda_N \quad (33) \qquad \lambda_{D4} = \frac{1}{4} * R_{S4} \Rightarrow \frac{1}{32} * \lambda_N \quad (34)$$

## III.5. Computing the Reliability using the Generating Function

For any k-out-of-n system, the reliability could be computed using a generating function that is a polynomial as shown below [7,8]:

$$g_n(z) = \prod_{i=1}^{n} (q_i + p_i z) = \sum_{i=0}^{n} R_e(i,n) z^i \quad (35)$$

where $q_i$ is the unreliability of component $i$ and $p_i$ is the reliability of component $i$. and $R_e(i,n)$ is the probability that exactly $i$-out-of-$n$ components are working.

By substituting the node reliabilities (defined in equations (4) and (5)) into the equation above, it follows that:

$$p_i(t) = \mathrm{e}^{-(\lambda_N t)} \tag{36}$$

$$q_i(t) = 1 - \mathrm{e}^{-(\lambda_N t)} \tag{37}$$

$$g_n(z) = \prod_{i=1}^{n} (1 - \mathrm{e}^{-(\lambda_N t)} + \mathrm{e}^{-(\lambda_N t)} z) = \sum_{i=0}^{n} R_e(i,n,t) z^i \tag{38}$$

where $z$ is a dummy variable, $R_e(i,n,t)$ is the coefficient of term $z^i$ in the polynomial and represents the probability that exactly i-out-of-n nodes are working at time $t$.

In order to get the reliability of the whole network then:

$$R(k,n,t) = \sum_{i=0}^{n} R_e(i,n,t) z^i \tag{39}$$

where $R(i,n,t)$ is the coefficient of term $z^i$ in the polynomial and represents the total reliability of the system or the probability that at least i out of the n nodes are working.

Figure 14 shown below shows a graph of $R_e(i,n,t)$ for a cluster of ten nodes. As shown at time zero, there is a probability of one that ten out of ten nodes are working. The probability eventually decreases and the probability of having less nodes increase. Figure 15 shows $R(k,n,t)$ for the same cluster. $R(k,n,t)$ is considered a more informative

function than $R_e(i,n,t)$ as it shows the total reliability of the system. For example, by using the data in Figure 15, the probability of having at least 10 nodes for any point of time can be extracted which is equal to the reliability of the cluster or the average number of nodes alive at any point in time.

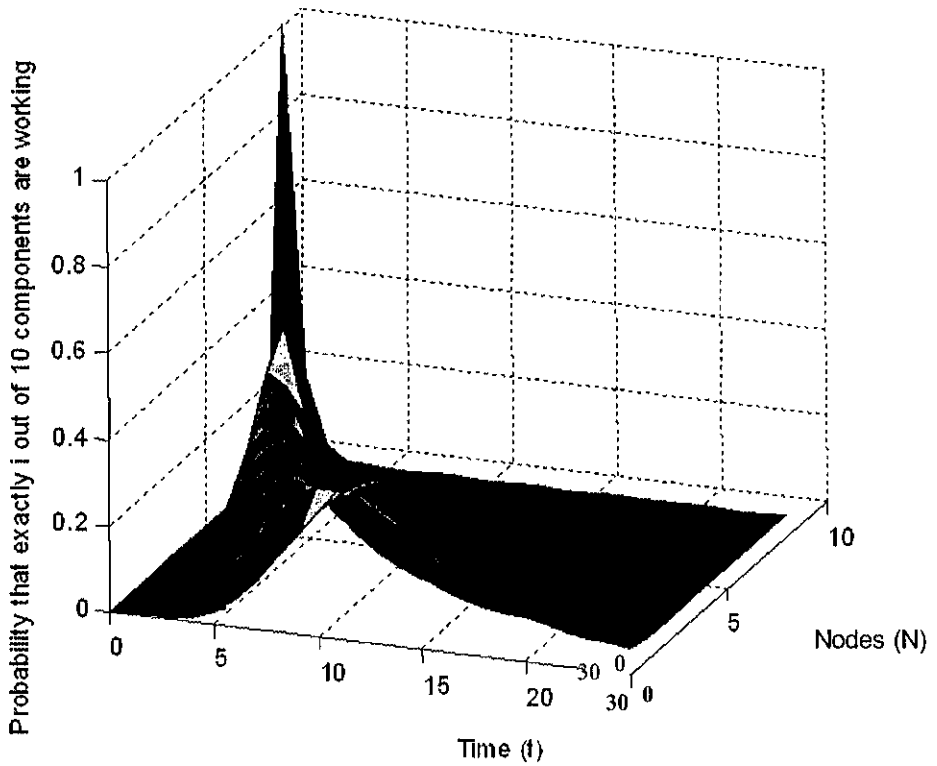



**Figure 14: 3D graph of Probability that exactly *N* out of 10 nodes are working at any time *t*.**

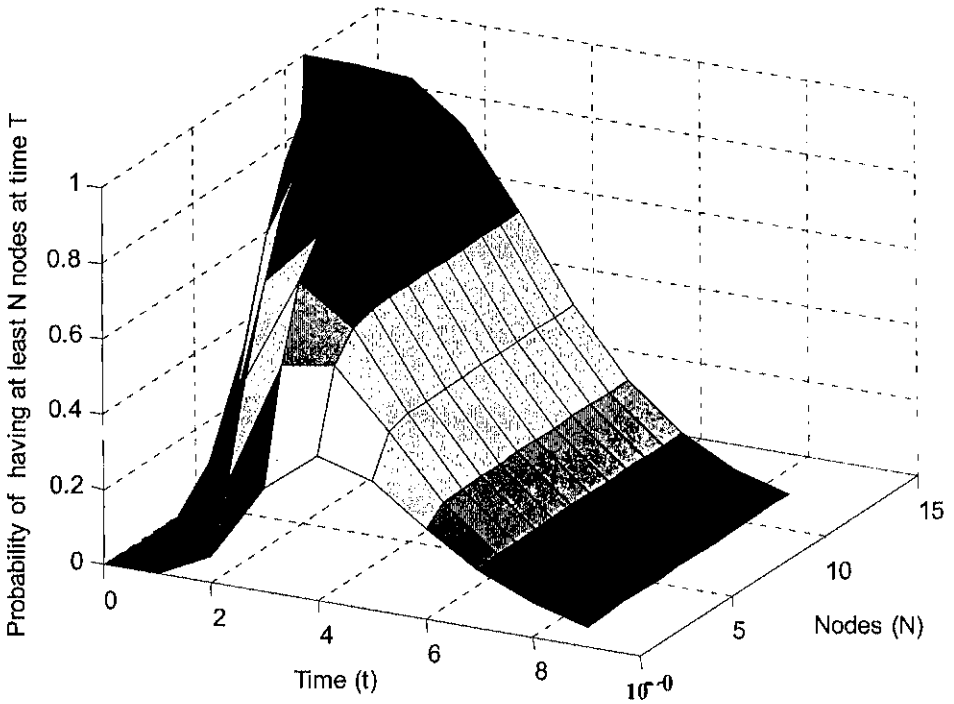



**Figure 15: 3D graph of Probability that at least N out of 10 nodes are working at any time t .**

# IV. RESULTS

## IV.1. Simulation

In order to verify the theory presented in Chapter III, a simulation of the wireless sensor network was performed using C++ and Matlab. The network model from Section III.3 was used with the following parameters:

- The network consists of a total of four hundred nodes
- An arrival rate of five tasks per time unit to the sink was used. Each task targets a random cluster with a random QoS value ranging from one to four nodes
- A task consists of sensing some variable $y$ of the environment. As a result, each sensor node is assigned with a random variable $y$ which it reports
- The training protocol consists of four coronas and eight wedges (thirty two clusters)
- The number of bits in a message is assumed to be 20K bits. Each message traverses a distance of 20m.
- Each node has an initial energy level of 5J [10]. Once a node has depleted its energy then it is considered to be failed

The simulation can be divided into three main parts which are the sensor nodes, the actual network, and a statistical module which collects the energy levels and details of the node at every time unit. A node could either be a normal sensing node or a cluster head node. Once a node gets elected as a cluster head, then its main functionalities change. Figure 16 shows a use case diagram of the simulation with the three main actors involved: cluster heads, sensor nodes, and the sink.

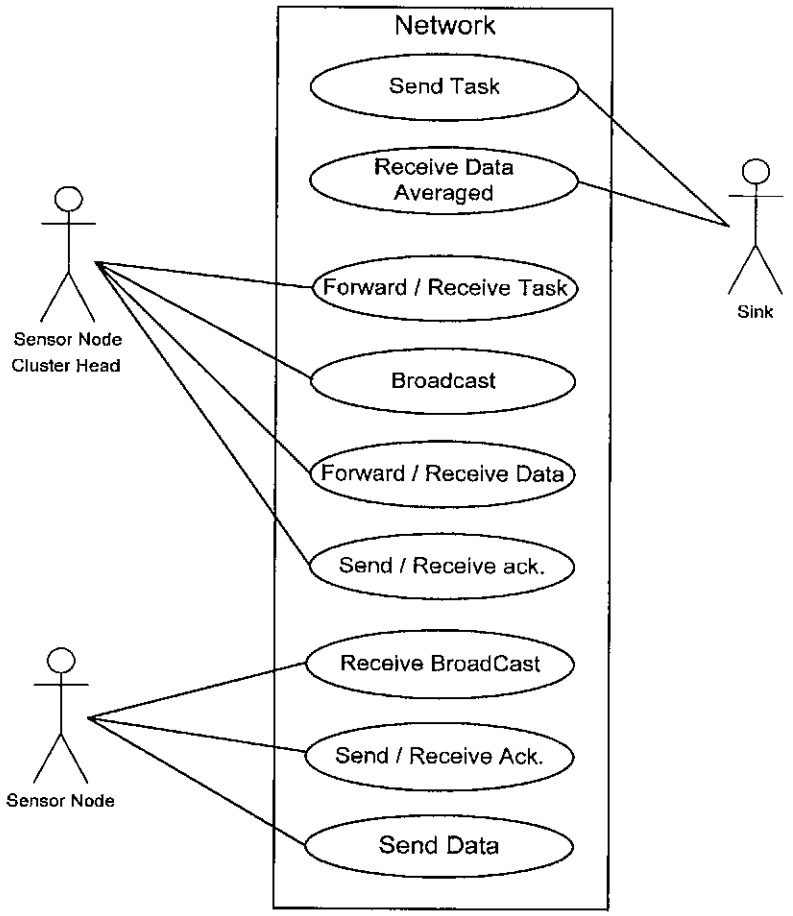



**Figure 16: Use case diagram for the simulation.**

A series of events occur when a cluster head receives a message. The sink node starts by sending tasks with a rate of five per time unit. Each task is randomly targeted to a certain cluster. When the task reaches its targeted cluster, the cluster head broadcasts it to the nodes within the cluster. In order to avoid collisions, a TDMA scheme is used where the cluster head opens a time window and waits for the nodes to reply. Nodes that are willing to work in the task will send a reply back to the cluster head, which is referred to as an "interest". The cluster head will then choose the first nodes that reply within the time window. It will choose a number of nodes large enough to satisfy the QoS requirement. The chosen nodes will then receive an "ack" (acknowledgment) message

from the cluster head. Nodes that receive this "ack" are now dedicated to the task. The acknowledged nodes will sense the data needed and send it to the cluster head. This series of events is shown in Figure 17. Once the cluster head receives all the data from the nodes, it will forward it to the lower level cluster. The clusters of the first corona will send the data to the sink.

The energy model described in Section III.3.1 was used. The values of $\alpha_r$, $\alpha_t$, and $\alpha_{amp}$ were taken as 50nJ/bit, 50nJ/bit, and $100 pJ/bit/m^2$ respectively **[6]**. Using equations (11) and (12) , the energy parameters could be calculated:

$$E_{tx} = 0.045\,\mathrm{J} \tag{40}$$

$$E_{rx} = 0.025\,\mathrm{J} \tag{41}$$

The energy lost for each message could be found by dividing each task and forward into simple events and calculating the energy consumed for each event. Figure 17 shows the series of events explained earlier that occur when a cluster receives a task and the associated energy consumed with each event. In the event of "*Nodes Receive + Reply*", all the nodes receive the broadcast, and then send back an "ack". Therefore, more than one node participates so the energy consumed multiplied by the total number of nodes involved. The same occurs in the events "*Nodes Receive Ack*" and "*Nodes Send Data*". Since the number of nodes that are involved in these events is dependent on the QoS value, the energy consumed will be multiplied by the QoS.

| Action | CH Recieves → CH Broadcasts → Nodes Receive + Reply → CH Sends Ack → Nodes Receive Ack → Nodes Send Data → CH Recieves Data |
|---|---|
| Energy Used: | $E_{rx} + E_{tx} + \begin{bmatrix}(\text{Nodes}-1)*E_{tx}\\(\text{Nodes}-1)*E_{rx}\end{bmatrix} + E_{tx} + \text{QoS} * E_{rx} + \text{QoS} * E_{tx} + E_{rx}$ |

**Figure 17: Events of a task on a cluster.**

The total energy consumed across the whole cluster for every task message would then be the sum of each individual event. Since all tasks are uniformly distributed and the cluster head is randomly chosen in a uniform fashion, then all nodes in the same cluster will consume energy with the same rate. Also, the node failures per job should be defined at the node level, therefore, the total energy lost must be in terms of each node rather than each cluster. Therefore, the total energy consumed must be divided by twelve to change it to the node level (twelve nodes per cluster). To get the fraction of total energy dissipated per node then it must also be divided by 5J that is the initial energy level in a node.

$$\frac{\text{Node Failures}}{\text{Jobs (Tasks)}} = \frac{\text{Energy Used (node)}}{\text{Task}} = \frac{2E_{rx} + 2E_{tx} + QoS(E_{rx} + E_{tx}) + (Nodes - 1)(E_{rx} + E_{tx})}{60} \tag{42}$$

For forwarded messages, nodes incur two events that are receiving the message and forwarding it. Therefore, the fraction of the total energy of a node used due to a forward task is

$$\frac{\text{Node Failure}}{\text{Jobs (Forwards)}} = \frac{\text{Energy Used by a node}}{\text{Forward}} = \frac{E_{rx} + E_{tx}}{5} \tag{43}$$

## *IV.2. Analytical Results*

The analytical results must be calculated using the equations developed in Chapter III and compared against simulation results. The main goal is to calculate the reliability of each cluster by finding its failure rate. In order to find the failure rate then the energy consumption and messages arrival must found first.

Assuming an arrival rate of $\lambda_N$=5, Table 3 shows the summary of the different arrival rates in each cluster. The arrival rate for forwarding events can be defined by

using equations (24) to (26) which is shown in the third column of the table. Similarly the arrival rates of messages are calculated using equations (27) through (34) .

**Table 3: Analytical Results of Arrival Rates**

| Messages Arrival Rate | Tasks Arrival Rate | Forwards Arrival Rate |
|---|---|---|
| $R_{S1} = \frac{5}{8}$ | $\lambda_{D1} = \frac{5}{32}$ | $\lambda_{F1} = \frac{20}{32}$ |
| $R_{S2} = \frac{15}{32}$ | $\lambda_{D2} = \frac{5}{32}$ | $\lambda_{F2} = \frac{15}{32}$ |
| $R_{S3} = \frac{5}{16}$ | $\lambda_{D3} = \frac{5}{32}$ | $\lambda_{F3} = \frac{5}{32}$ |
| $R_{S4} = \frac{5}{32}$ | $\lambda_{D4} = \frac{5}{32}$ | |

$N_{D1}$ and $N_{F1}$ are defined as the average number of tasks and forwards (respectively) a node can perform before failing. Therefore, they are the inverse of equations (42) and (43) respectively. The average QoS value used in the simulation was two. The first and second column of Table 4 show the average energy consumed in the clusters.

The failure rates for the clusters are calculated by substituting the values from Tables 3 and 4 into equations (15) to (18).

**Table 4: Analytical Results of energy loss**

| Energy loss- forwards | Energy loss - Tasks | Failure Rates of Clusters |
|---|---|---|
| $N_{F1} = 87.46$ | $N_{D1} = 140.84$ | $\lambda_{cluster1} = 0.0106$ |
| $N_{F2} = 77.22$ | $N_{D2} = 140.84$ | $\lambda_{cluster2} = 0.0086$ |
| $N_{F3} = 71.98$ | $N_{D3} = 140.84$ | $\lambda_{cluster3} = 0.0066$ |
| $N_{F4} = 69.28$ | $N_{D4} = 140.84$ | $\lambda_{cluster4} = 0.0045$ |

By substituting the values of the failure rates in equation (9), the reliability distribution for each cluster can be found.

$$R_1(t) = \prod_{i=1}^{m} (1 - e^{-0.0106t}) \tag{44}$$

$$R_2(t) = \prod_{i=1}^{m} (1 - e^{-0.0086t}) \tag{45}$$

$$R_3(t) = \prod_{i=1}^{m} (1 - e^{-0.0066t}) \tag{46}$$

$$R_4(t) = \prod_{i=1}^{m} (1 - e^{-0.0045t}) \tag{47}$$

## *IV.3. Simulation Results*

This section presents the results of the simulation. The statistical module in the C++ simulation was responsible for collecting the data such as the number of nodes alive in each cluster and corona level at every time unit. The simulation was run 1,000 times where each run lasted for 600 time units. The data was then averaged over the number of runs and imported into Matlab from which the graphs were obtained. Because of the symmetry of the network, each wedge is statistically independent from the rest. Therefore, we can compute an improved estimate by averaging the results from the eight wedges. For example, all the data obtained from the clusters of the first corona was averaged over eight to get the average data of a cluster in the first corona level.

The plots in this section are compared and plotted against the analytical results from the previous section for each corona level. The main parameter of analysis used is the reliability and unreliability of a cluster in each corona level that is determined by the number of nodes alive in a cluster.

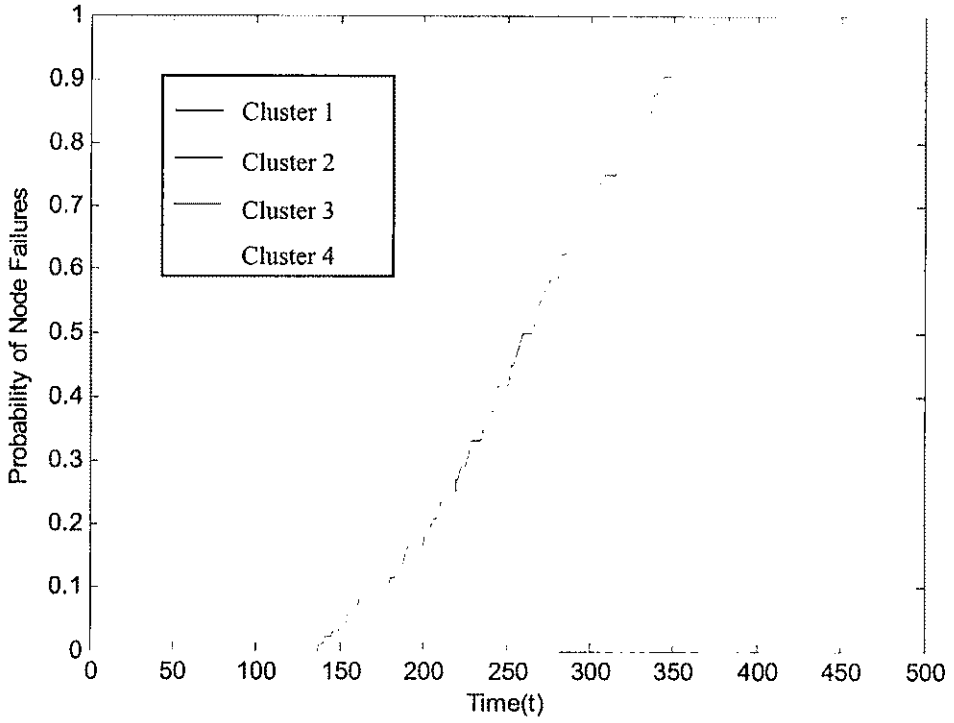



**Figure 18: Performance of all four clusters.**

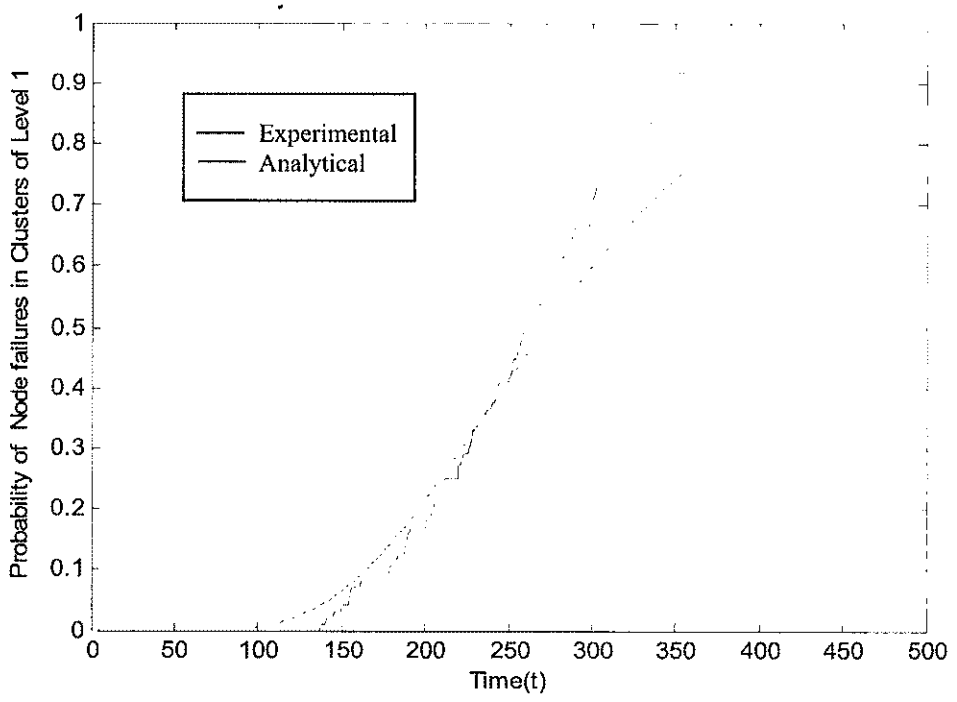



**Figure 19: Results of Cluster from the first corona.**

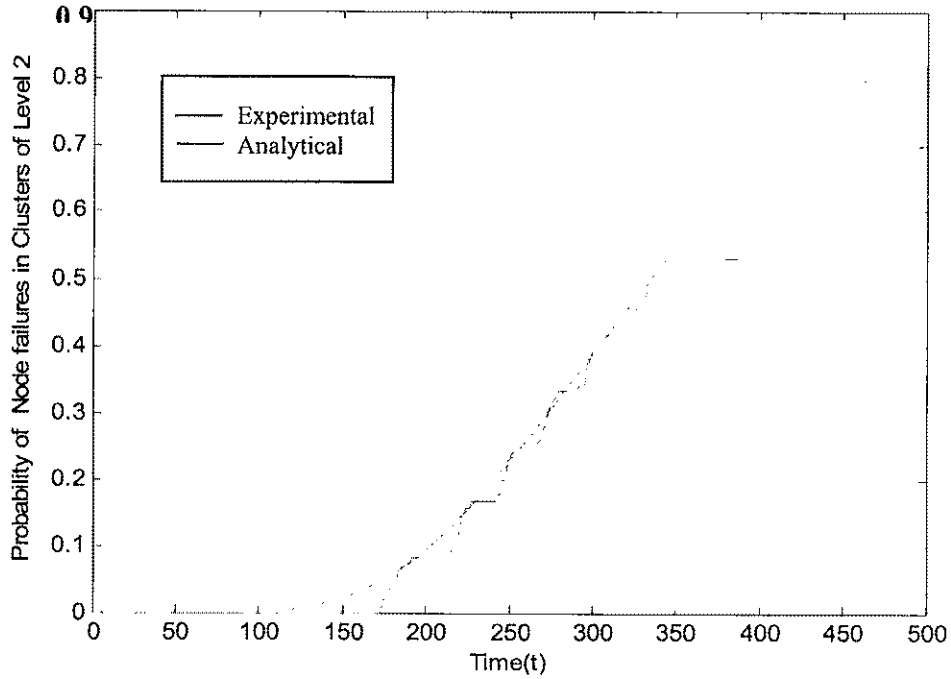



**Figure 20: Results of Clusters from the second corona.**

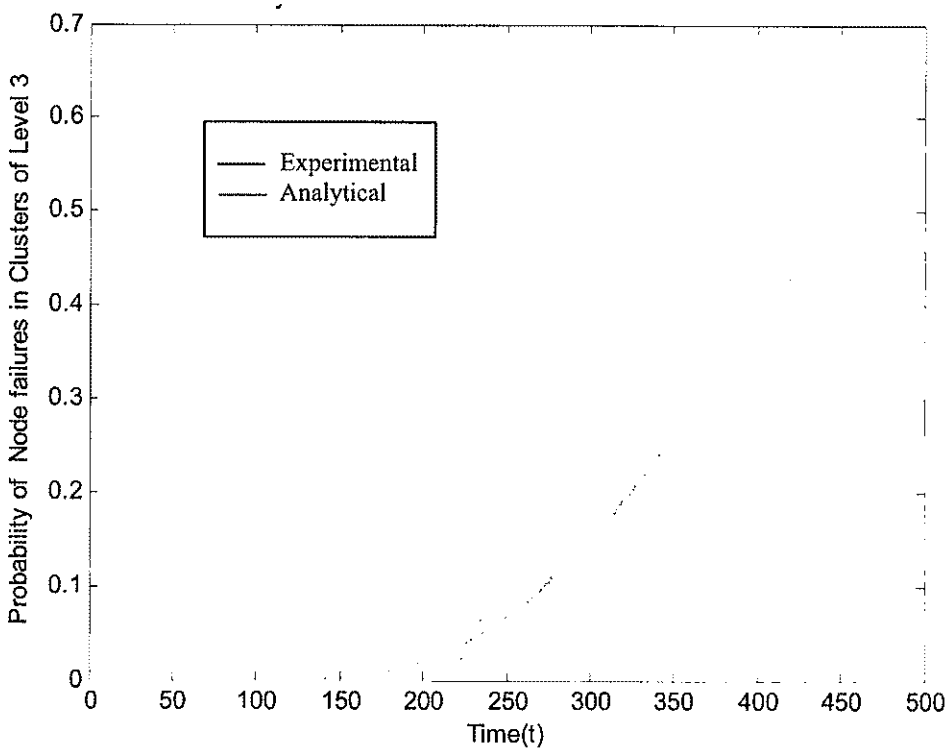



**Figure 21: Results of Cluster from the third corona.**

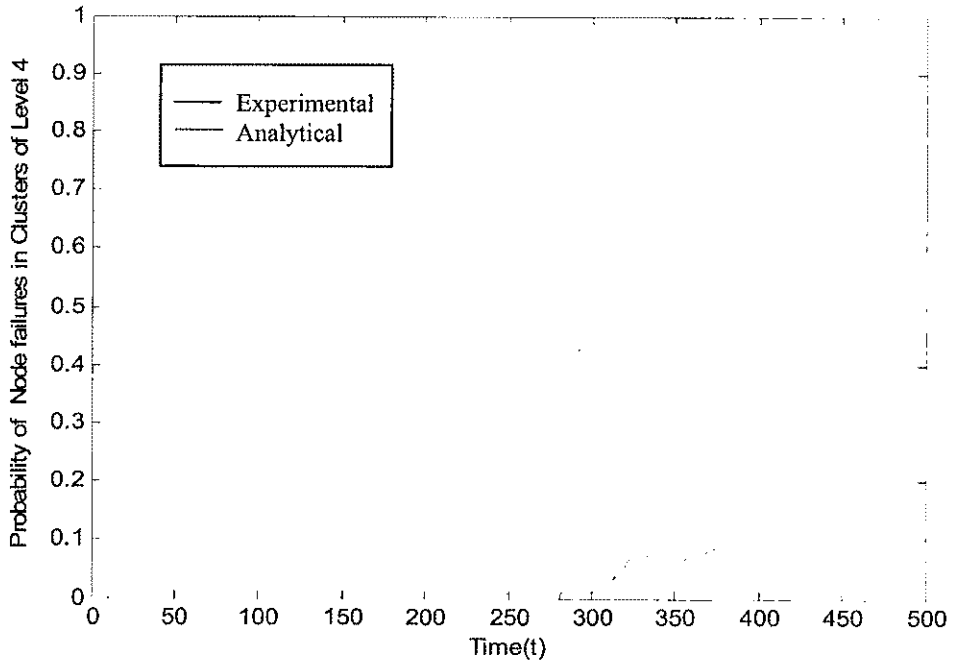



**Figure 22: Results of Clusters from the fourth corona.**

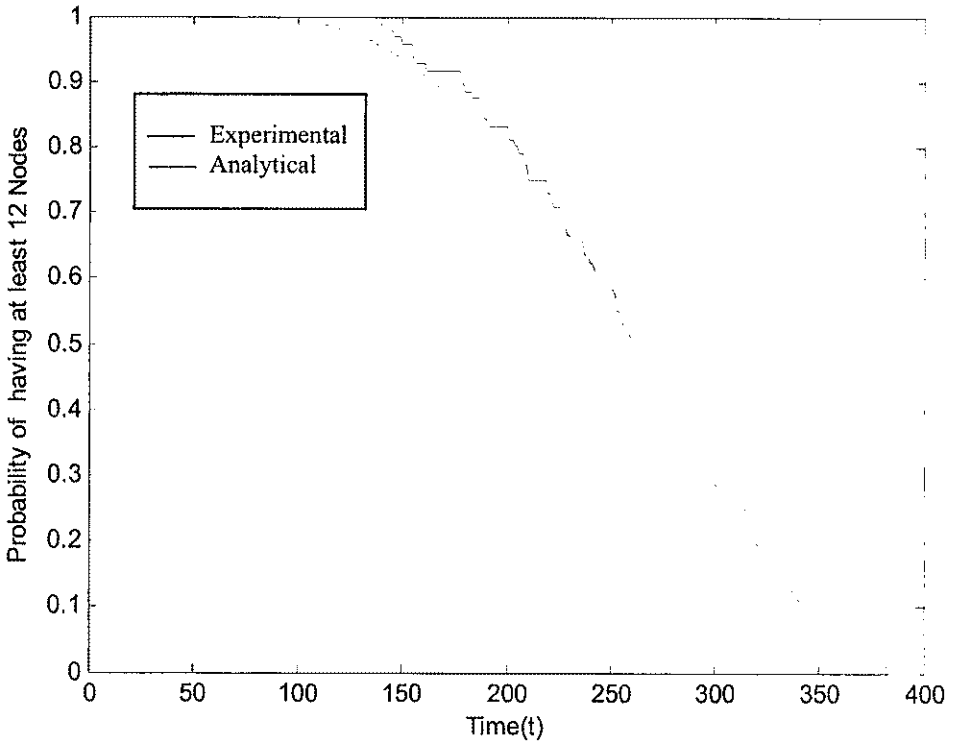



**Figure 23: Results of generating function for clusters from the first corona.**

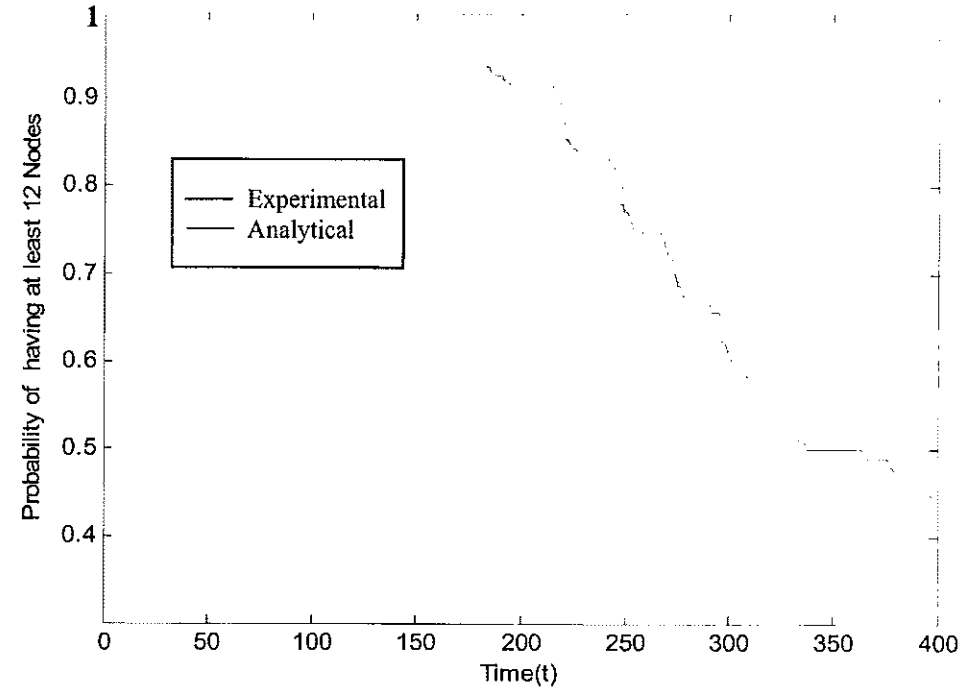



**Figure 24: Results of generating function for clusters from the second corona.**

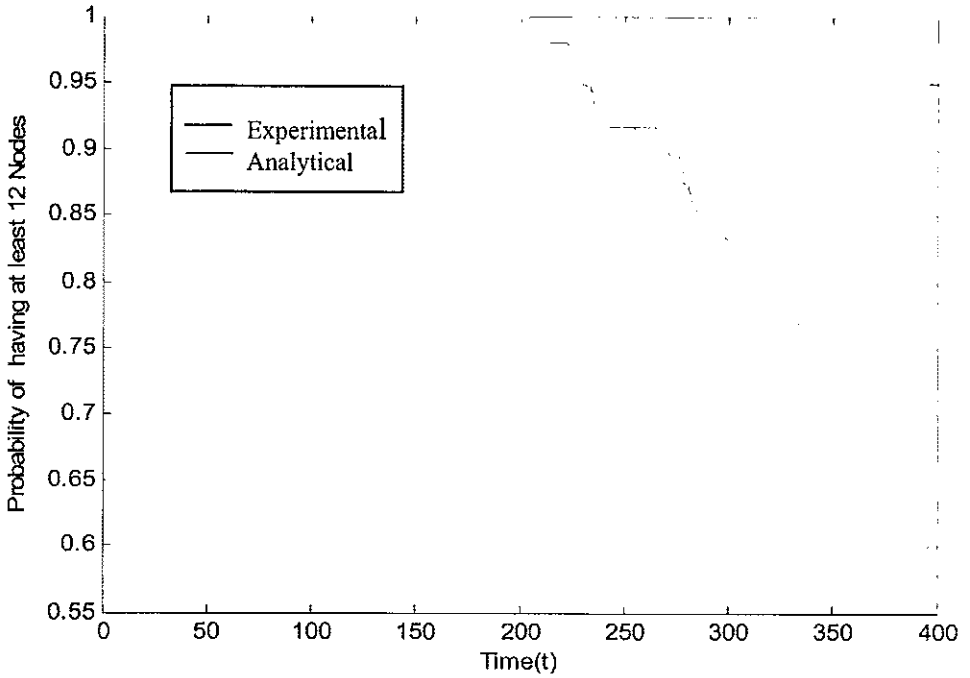



**Figure 25: Results of generating function for clusters from the third corona.**

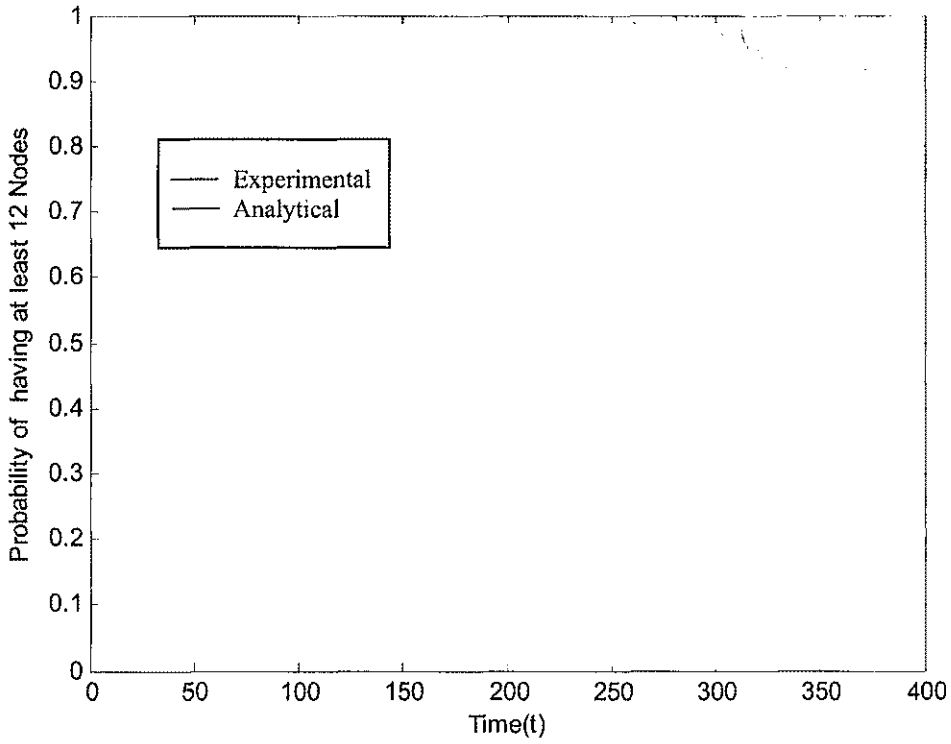



**Figure 26: Results of generating function for clusters from the fourth corona.**

The four clusters studied in this work are shown in Figure 18. The first cluster fails after 350 time units. Therefore, according to equation (10) the analytical model is valid from:

$$0 \langle t \langle 3\,5\,0 \tag{48}$$

The failure rate decreases for clusters that are farther from the sink. The first cluster experiences the highest failure rate. The rate decreases through the second and fourth cluster. This same trend occurs for the time until the cluster experiences its initial failures. The first cluster starts to experience node failures after 140 time units, the second cluster after 170 time units, the third cluster after 205 time units, and the fourth cluster after 280 time units.

As shown, the failure rate for all of the four clusters has a step structure. A node in a cluster fails instantly causing the failure rate to increase discretely. This instant change of the failure rate causes the step structure to form. For the first cluster, there are twelve steps because of the twelve node failures that occur in the cluster for the duration in which the analytical model is defined. As the number of node failures decreases for the upper clusters, the number of steps also decreases.

Figure 19 through Figure 22 shows a comparison between the simulation data and the analytical results for the four different clusters. The analytical plots are obtained from equations (44) to (47). There is a close agreement between the simulation and analytical results up to 250 time units, after which the probability of the simulation data fails at a higher rate than the analytical. The results of the second cluster are shown in Figure 20 where the analytical and experimental results closely follow the same trend. Figure 21 shows the results from the third cluster that shows similar behavior. As for the fourth cluster, the two data sets have the same trend.

The second sets of data collected are shown Figure 25 through Figure 23. The analytical results are obtained from equations (38) and (39) by only considering $R(12, n, t)$. This gives the probability of having at least 12 nodes in a cluster that is considered to be the reliability of each cluster. For the first cluster, the analytical and experimental results closely match until 250 time units and then partially differ after that. There is a good quantitative comparison between the simulation and analytical results for the second and third cluster. The fourth cluster shows a good qualitative comparison.

The last and most important data set is shown in Figure 27. This experimental plot represents the total probability of node failure in the entire network (all four clusters combined). The analytical plot is obtained from equation (23) that is the failure rates of the entire network. As shown, the analytical model accurately matches the experimental results.

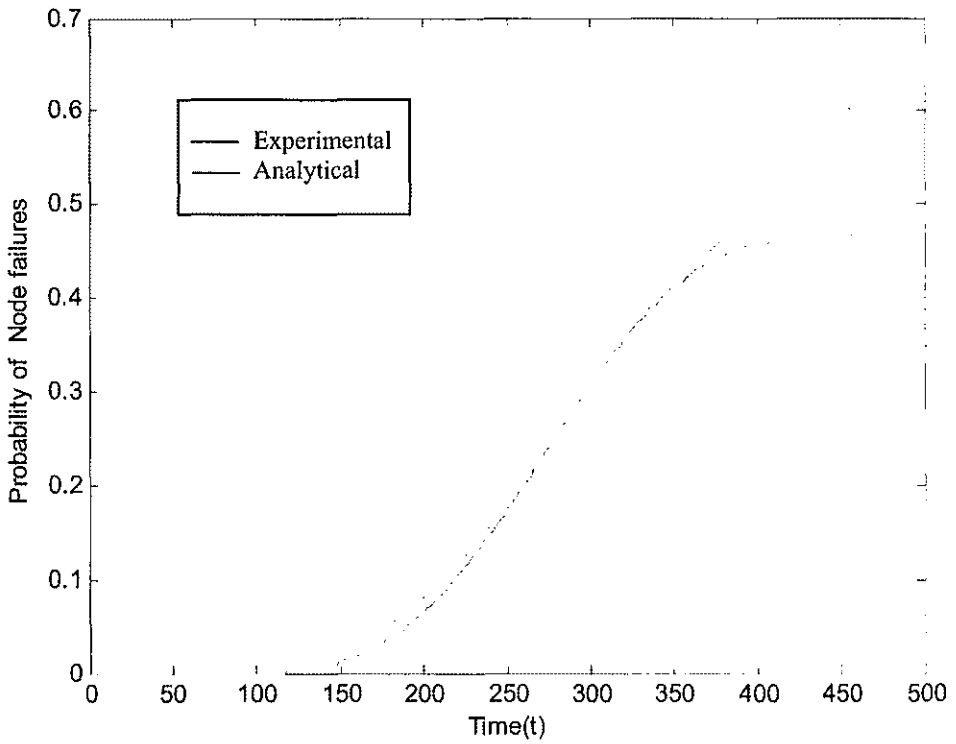



**Figure 27: Results of the average of the four clusters.**

The first cluster is the busiest cluster since it is the closest to the sink. Therefore, more tasks and forwards are being received from the upper clusters and the sink at random times, which makes it more difficult to model. This results in the slight difference between the analytical and simulation results in the first cluster. As for the fourth cluster, it is at the outmost corona level and receives the least amount of task and forward

messages. Therefore, the energy consumption in it is very low and not much node loss occurs in it.

The simulation results and analytical model closely match for the defined network time. By the time the simulation results and analytical data diverge, the network is in a pathological state because once the first level clusters have failed, then the upper clusters can no longer communicate with the sink. As a result, after the diverge, the reliability of the nodes reaches a constant value because the clusters are not receiving any new tasks from the sink and all the forward messages sent to the clusters of the first corona are lost.

# V. FUTURE WORK

This section describes some possible directions for future work. This work has revealed many uncovered areas in wireless sensor networks. Now that the basic reliability and fault tolerance framework has been designed for one type of network, much further research can evolve in this area.

First of all, the network model can be further enhanced. A more accurate network model could include clusters with different node densities i.e. a different number of nodes in each cluster. The network that was studied in this work failed once the clusters of the first corona failed. Therefore, the reliability of the network could be studied to see whether having a greater number of nodes would dramatically increase the network lifetime and reliability or not. Furthermore, the workload distribution can be studied to observe the effect of having certain clusters with a higher workload on the network reliability. A network where clusters in different wedges can communicate together can also be studied. Such a model would result in traffic going across the coronas and not only across wedges.

By having a valid reliability function, then all the parameter tradeoffs could be initially studied before deployment. Such parameters are the different arrival rate of jobs to the sink, the node density functions, the node transmitter range and the nodes energy level.

Secondly, different failures such as adversary attacks could be studied. For example, the effect of an adversary node attempting to flood the network by excessive messages could result in a higher failure rate of surrounding nodes and a network

congestion. This case could be studied by considering the reliability functions of the nodes and the different tradeoffs required to neutralize the adversary.

Finally, an actual deployment of the network could result in more accurate results. The results obtained in this work were through accurate simulations; however, a simulation can never be perfectly accurate. An actual deployment might reveal hidden characteristics of the network reliability.

# VI. CONCLUSIONS

The goal of this work was to model the performance and reliability of a wireless sensor network. A generic fault tolerance framework was given that models the different stages a wireless sensor network experiences due to node failures.

An extensive analytical model was derived to study the failure rates of the different components in the network. The node's reliability distribution was modeled using a Poisson distribution. The failure rates of the nodes due to energy and physical failure were also modeled. The energy failure rate was modeled in terms of the arrival of the jobs to each node and the energy consumption of each job.

By studying the dynamics of the network, the different arrival rates for jobs and forwards were defined for each cluster. A valid sensor energy model was given. The failure rate of the nodes for each cluster in the network was expressed in terms of the different arrival rates and the energy consumption of jobs.

The network's failure rate was then studied by combining the failure rates of each cluster. An alternative reliability model was also given which is obtained from a simple polynomial expression and provides a three dimensional representation of the system in terms of reliability, time and nodes.

To verify and prove the integrity of the reliability model derived, a simulation of the network was performed using C++. The simulation's goal was to model the networks performance by using a valid energy model for each node and measuring the node losses and energy level periodically. To achieve accurate results, a constant work load was used which targets random clusters. The simulation was run for 1,000 times and the results were averaged. A comparison of the analytical model and the simulation results was

given, which shows a strong match between the analytical model and the simulation results for the duration that clusters of the first corona are working. After they fail, the network is considered to be down as the sink can no longer communicate with the upper level clusters.

To sum up, a valid reliability model was designed which represents the node losses due to energy consumption in the network. This model can be used to measure the fault tolerance of the network to different parameters before actually deploying the network. An estimate of the network's performance can also be derived using the analytical model introduced in this work that could help and save time in the design and deployment of the network.

# REFERENCES

[1] A. P. Speer, I. Chen, "Effect of Redundancy on Mean Time to Failure of Wireless Sensor Networks," AINA pp. 373-382, 20th International Conference on Advanced Information Networking and Applications - Volume 2 (AINA'06), 2006.

[2] A. Wadaa, S. Olariu, L. Wilson, K. Jones, Q. Xu, "On Training a Sensor Network," *ipdps*, p. 220b, International Parallel and Distributed Processing Symposium (IPDPS'03), 2003.

[3] L. J. Bain and M. Englehardt,, Statistical Analysis of Reliability and Life-Testing Models: Theory and Methods, 2nd ed., Marcel Dekker, New York, 1991.

[4] A. Hoyland and M. Rausand, System Reliability Theory: Models and Statistical Methods, vol. 518. John Wiley and Sons, 1994

[5] R. Szewczyk, J. Polastre, A.M. Mainwaring, D.E. Culler. Lessons from a Sensor Network Expedition. Proc. of 2nd IEEE European Workshop on Wireless Sensor Networks (EWSN2004).

[6] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan,,Energy-Efficient Communication Protocol for Wireless Microsensor Networks," Proc. of the 33rd Annual Hawaii International Conference on System Sciences(HICSS), pp. 3005=3014, January 2000

[7] R. E. Barlow and K. D. Heidtmann, "Computing k-out-of-N Reliability," IEEE Trans. on Reliability, vol. R-33, no. 4, pp. 322-323, October 1984.

[8] W. Kuo and M. J. Zuo, Optimal Reliability Modeling: Principles and Applications Hoboken, NJ: John Wiley & Sons, 2002.

[9] W. B. Heinzelman, A. L. Murphy, H. S. Carvalho, and M. A. Perillo. Middleware to Support Sensor Network Applications. IEEE Network,18(1): 6.14, January - February 2004.

[10] G. Gupta and M. Younis, Fault-Tolerant Clustering of Wireless Sensor Networks, IEEE Wireless Communications and Networking, 3:1579 – 1584, March 2003.

[11] M. Staroswiecki, G. Hoblos, and A. Aitouche, "Fault tolerance analysis of sensor systems," in Proc. 38th IEEE Conf. Decision and Control (CDC'99), Phoenix, AZ, 1999.

[12] G. Hoblos, M. Staroswiecki, and A. Aitouche, "Optimal Design of Fault Tolerant Sensor Networks," Proc. of the 2000 IEEE International Conference on Control Applications, Anchorage, Alaska, pp. 467–472, September 2000.

[13] G. Hoblos, M. Staroswiecki, A. Aitouche, "Sensor network design for fault tolerant estimation" in Proc. 38th IEEE Conf. Decision and Control (CDC"99) Phoenix, AZ, pp 55-72, 1999.

[14] C.-F. Chiasserini and M. Garetto. Modeling the performance of wireless sensor networks. In Proceedings of IEEE Infocom, pp 220-231, March 2004

[15] D. S. Kim; Shazzad KM; J. S. Park. A framework of survivability model for wireless sensor network; The First International Conference on Availability, Reliability and Security, ARES 2006. 20-22 April 2006.

[16] T.-Y. Wang, Y. S. Han, P. K. Varshney and P.-N. Chen, "Distributed Fault-Tolerant Classification in Wireless Sensor Networks," IEEE Journal on Selected

Areas in Communications (JSAC): special issue on Self-Organizing Distributed Collaborative Sensor Networks, pp. 724-734, April, 2005.

[17] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentell. "Fault Tolerance Techniques for Wireless Ad Hoc Sensor Networks". vol 2, pp 1491.1496. Sensors, 2002, Procedings of IEEE, 2002.

[18] "Fault Tolerant Sensor Networks with Bernoulli Nodes", Chih-Wei, Peng-Jun Wan, Xiang-Yang Li, Ophir Frieder, IEEE Wireless Comm. and Networking Conf.

[19] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," IEEE Communications Magazine, 2002.

[20] Th. Arampatzis, J. Lygeros, S. Manesis, A Survey of Applications of Wireless Sensors and Wireless Sensor Networks, Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation, June 27-29 2005, pp. 719-724.

[21] M. Vieira, D. da Silva Jr., C.C. Jr., and J. da Mata, "Survey on wireless sensor network devices," in Proceedings of the 9th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'03), Lisbon, Portugal, Sept. 2003.

[22] K. S. Low; W.N.N. Win; M. J. Er," Wireless Sensor Networks for Industrial Environments", Computational Intelligence for Modeling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, vol. 2 pp 271- 276 November 2005

[23] I.Demirkol, C.Ersoy, F.Alagoz, "MAC protocols for wireless sensor networks: A survey", IEEE Commun. Mag., vol.44, no.4, pp.115-121, 2006/04.

[24] B. Warneke, M. Last, B. Leibowitz, and K.S.J. Pister. Smart Dust: communicating with a Cubic-Millimeter Computer. IEEE Computer, pp 44-51, January 2001.

[25] J. N. Al-Karaki,, and A. E. Kamal. Routing techniques in wireless sensor networks: A survey. , IEEE Wireless Communications, vol. 11, no. 6, pp. 6-28. 2004

[26] R. Billinton and R. N. Allan, Reliability Evaluation of Engineering System: Concepts and Techniques, Perseus Publishing, 1992.

[27] B. Krishnamachari and S. Iyengar, "Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks," IEEE Trans. Computers, vol. 53, no. 3, pp. 241-250, March 2004.