A CONSENSUS PROTOCOL FOR WIRELESS SENSOR NETWORKS

by

MUKUL KUMAR

THESIS

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

2003

MAJOR: **COMPUTER SCIENCE**

Approved by:

_____

Advisor Date

ACKNOWLEDGMENTS

I thank Professor Loren Schwiebert for his immense help and support in my research. He has been the guiding factor and the torchbearer who has shown me the right direction every time I got lost in dark caverns of wireless sensors. He has in depth knowledge of the subject and his enthusiasm about the research motivated me in finishing my research in time. He helped me in my research by not only providing guidance but also providing reading materials and research papers in order to give proper direction to my research. I would state that the thesis would not have been possible without his umpteen guidance and I feel honored to have done my thesis under him.

Professor Brockemeyer, and the members of the NEWS lab provided me insight and helped me in not only resolving certain key issues but also in the implementation of the protocol. I appreciate their help in this regard.

In the end, I would like to thank my parents and my sister for their immense support and for being there when I always needed them. They have not only helped me financially but also emotionally during the past two years at Wayne State.

TABLE OF CONTENTS

Chapter **Page**

CHAPTER 1

INTRODUCTION

Advances in silicon technology have led to the development of next generation sensors. These sensors communicate wirelessly to transmit their readings and to reach an agreement or conclusion about their information. They are called wireless sensors and present a new facet in the field of communication and computer networks. Wireless sensors are compact devices that integrate communication, computation, and micro-electrical mechanical (MEMS) devices into a single chip [32].

The nodes consume extremely low power; have low communication rates, which are measured in kilobits per second, and potentially operate in high volumetric densities. The communication process is highly energy consuming and should thus be minimized [5]. Sensor networks are densely deployed in order to provide a sufficient amount of redundancy to the system [16]. The nodes are usually placed either near or inside a phenomenon. Redundancy in the system enables conservation of energy in the system by allowing some of the nodes in the system to be in the sleep state. This also leads to an improvement in fault tolerance and an increase in the operating life of the system. This improves the probability that all phenomena are observed and analyzed by the system. The reliability or fault tolerance of a system is modeled [17] using a Poisson distribution to capture the probability of not having a failure within the time interval of (0,*t*):

$$R_k(t) = \exp(-A_k, t) \quad (1)$$

Where A_k and t are the failure rate of sensor node k and the time period, respectively.

Wireless sensors are very robust and versatile, and are deployed in large groups in order to gather and disseminate information. These groups of wireless sensors work as autonomous units and form a wireless sensor network. A wireless sensor network is assumed to be a homogeneous network. It consists of multiple nodes communicating with the base station. A base station serves as the client or user end from which the user controls the wireless sensor network. The base station can be used to request information from the sensor network. The base station then receives information from the sensor network and processes the information obtained from the sensor network. The base station can be either mobile or stationary. The sink performs local data aggregation within the wireless sensor network. The wireless sensor network is designed to detect events and propagate the information back to the base station. The network can also be designed in such a manner that the base station ascertains information about the events in a particular region of the network. A query is sent out by the base station to request some information. The query is propagated to the region where the events are to be examined. The result is then sent back to the base station.

Wireless sensors usually operate in large groups. The density of nodes in a particular area of the sensor network can be approximated by the following formula [16,18]. This assumes the communication range of a sensor can be modeled as a circle i.e. no obstacles.

$$\mu(R) = (N \pi R^2) / A \qquad (2)$$

Where *N* is the number of scattered sensor nodes in region A; and R, the radio transmission range. $\mu(R)$ gives the average number of nodes within the transmission radius of each node in region *A*.

1.1 *Wireless Sensor Network Classification:*

Wireless sensor networks can be organized into different classes based on their applications. They can be effectively employed for Military, Environmental, Health, and Home Applications, etc. [16]. Biomedical sensor networks [34] entail the sensor nodes to be fixed, which enables the predetermined placement of the nodes in a system. The relay sensors are then employed in order to provide connectivity of sensor nodes to the base station [33]. Random placement of nodes is done where the position of nodes cannot be predetermined. Suitable applications for random placement include military and surveillance applications like determining forest fires, house intrusion, etc. Various approach like LEACH [36] and PEGASUS [25] have been proposed for data gathering and routing in sensor networks. These routing techniques can be broadly classified under the following categories:

1.1.1 *Direct Transmission:*

The basic function of a wireless sensor is to sense the required information and send it back to the base station. Each individual node can send back information directly to the base station. This approach is, however, not encouraged, as each node sending back information directly to the base station would lead to a lot of energy consumption as well as bandwidth contention in the network.

1.1.2 *Clustering Algorithms:*

A node is selected as a cluster head and all the nodes in the vicinity of the cluster head communicate with the cluster head. The cluster head serves as a central point for data aggregation and local decision-making. The cluster head beamforms the signal and communicates with the base station. LEACH (Low Energy Adaptive Clustering Hierarchy) is a clustering protocol. It utilizes randomized rotation of local cluster base stations (cluster heads) to evenly distribute the energy load among the sensors in the network [36].

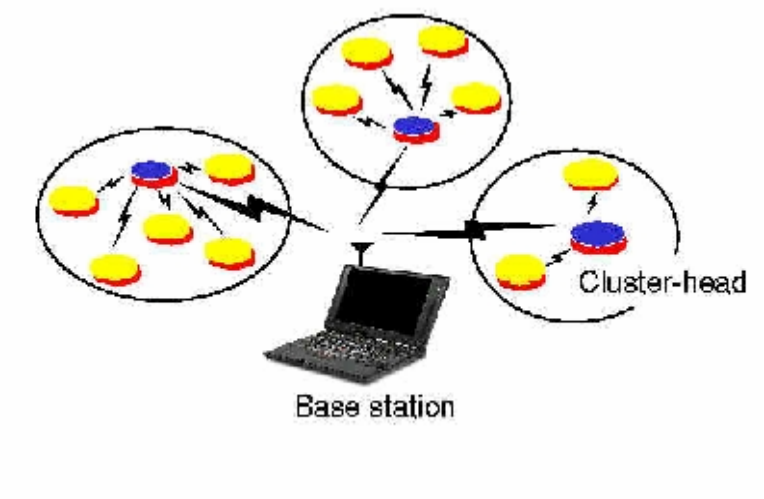

Figure I: An example of the cluster based approach (Rabiner [36])

1.1.3 *Hierarchical Clustering Algorithms:*

Multihop networks may employ one or more intermediate nodes in order to transfer messages from a node to the base station. Large sensor networks have groups of nodes with different information. Each group can be considered as a cluster and information from different clusters can be combined in different layers to form a hierarchy [37]. A hierarchical control structure is developed for multihop networks as shown in the Figure II.

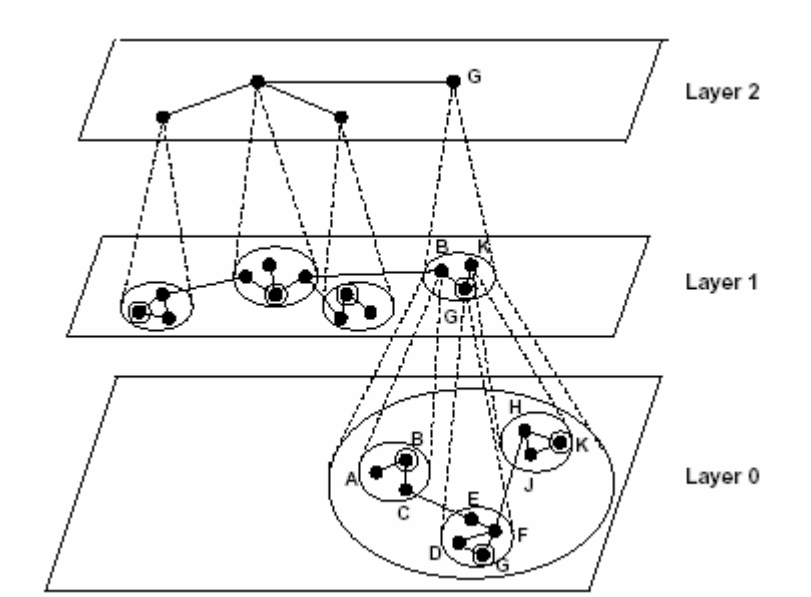

Figure II: An example of a three-layer hierarchy (Suman [37])

1.1.4 *Chain Based Algorithms:*

Sensor nodes form a chain by communicating with their closest neighbors. The information gathered moves from node to node, gets fused, and is eventually transmitted to the base station. PEGASIS [25] employs a chain-based algorithm in which the nodes take turns transmitting data to the base station. This enables load balancing by minimizing the energy spent by each node per round thereby increasing the life of the network.

Figure III shows an example of the chain-based approach. Nodes c0, c1, c2, c3, and c4 form a chain and node c2 transmits data back to the base station (BS).

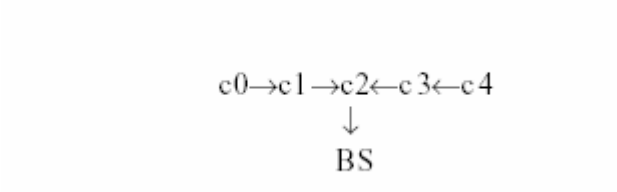

Figure III: Chain Based Approach (Lindsey [25])

1.1.5 *Directed Diffusion:*

Directed diffusion is a data centric model for routing information [4]. It utilizes the concept of area of interest, where each node has a specific area of interest, and the nodes are identified based on their area of interest. An interest from the sink is diffused to the nodes in the region where nodes have similar areas of interest. The sensors reply back to the base station with the required information. The intermediate nodes perform data aggregation and also improve the accuracy of information. The intermediate nodes can also cache, or transform data, and may direct queries based on previously cached data. The path from the nodes to the sink is reinforced and data delivery eventually takes place along the reinforced path.

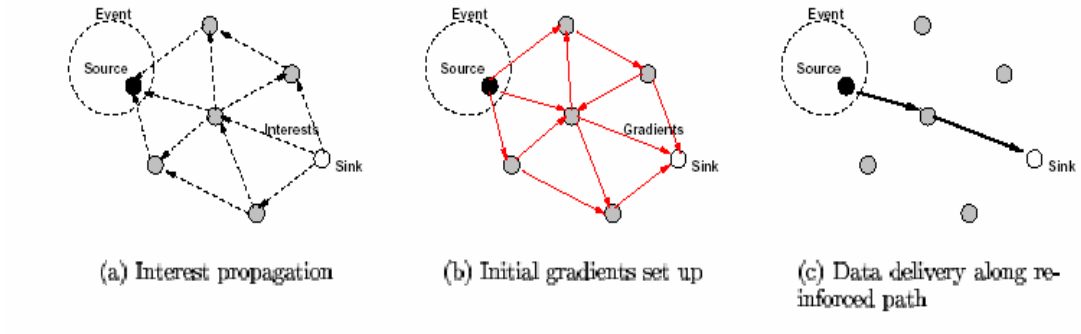

Figure IV: A simplified diagram for Directed Diffusion (Intanogonwiwat [4])

1.2 *Wireless Sensor Network as Distributed System*

The main motivation for constructing distributed systems is sharing of resources. Wireless sensors in a sensor network distribute the task of observing a phenomenon amongst various nodes based on the location or area of interest of the nodes. The nodes then share the information amongst themselves in a wireless sensor network. A wireless sensor network thus constitutes a distributed system. The design of a wireless sensor network should therefore be in conformance with the principles of distributed systems. Factors like fault tolerance, period of operation, and scalability should be considered while designing a wireless sensor network distributed system.

1.2.1 *Failures in Distributed System*

Distributed systems are usually designed to be fault tolerant. Faults can be classified into three categories: omission failures, arbitrary failures, and timing failures.

Omission Failure: Omission failures are encountered due to failures in process and communication channels [38]. A process omission failure in the sensor network takes place when a sensor node stops functioning or crashes. A

communication omission failure happens when the data being sent from node *P* to node *Q* is transmitted from node *P* but is never received at node *Q*.

Arbitrary Failure: An arbitrary or Byzantine failure of a process is one in which it arbitrarily omits intended processing steps or takes unintended processing steps [38].

Timing Failure: A timing failure occurs when the response of the communicated process takes more than the specified real-time interval [12].

1.2.2 *Period Of Operation:*

One of the most important criteria for the wireless sensor network is the longevity of operation. The wireless sensors should function correctly for a long period of time after their deployment. The wireless sensors derive energy from batteries present in the wireless sensor nodes. As the batteries usually have a fixed lifetime, it is thus desirable that the power consumption of the nodes should be minimized. Rechargeable batteries can also be employed for wireless sensors. They are attached to a solar or photovoltaic cell [39, 40]. The photovoltaic cells, however, produce a limited amount of power and therefore the energy must still be conserved. Communication costs account for the major amount of power consumption. Karayan and Potkonjak [2] mention that the ratio of power consumption for communication to computation is often estimated as 1000:1. Decreasing the amount of communication between nodes can reduce

the energy consumption and thus increase the life of the sensor network. A lot of research has therefore concentrated on minimizing the amount of communication taking place in the wireless sensor network.

Wireless sensors are deployed in large groups of hundreds or thousands of nodes in order to observe a phenomenon [16]. Protocols and sensor network solutions should thus be designed to be scalable to a large number of nodes without a corresponding increase in space complexity or communication costs. Localized algorithms have been proposed as an effective solution to the problem of scalability in large wireless sensor networks [4,36,37,41]. A fully distributed algorithm would further enhance the scalability of the sensor network.

Redundancy is introduced into the wireless sensor network in order to make the sensor network fault tolerant as well as increase the operating period of the network. As several nodes monitor the same area, some of the nodes can be put to sleep without any loss of precision in the network. This leads to conservation of energy in the network and thereby increases the lifetime of the network. Redundancy enables more than one node to observe phenomena in a sensor network. This improves the fault tolerance of the system as information from multiple nodes can be combined to obtain correct results.

1.3 *Consensus Generation In Wireless Sensor Network*

Wireless sensors communicate amongst themselves as well as the base station. The communication amongst the nodes can be one-to-one, multicast, or broadcast. The nodes would require information about neighboring nodes for both one-to-one communication as well as multicast. This information constitutes the state information present at the node. The state information needs to be refreshed periodically to account for changes in the topology due to movement of wireless sensors or due to nodes encountering crash failures. A protocol design should minimize the state information of the nodes in the network. A broadcast method of communication should thus be employed in order to distribute common information amongst the nodes.

Following the observations made about the wireless sensor network and its characteristics, we have devised a protocol for dissemination of information from the sensor nodes to the base station. The wireless sensors locally generate consensus amongst themselves and then send this information back to the base station. This reduces the amount of information to be sent back to the base station, leading to an improvement in energy costs of the network thereby increasing the life of the network. The faulty nodes would be unable to generate consensus for their information and would thus be unable to send back their information to the base station. This is based on the assumption that the number of faulty nodes is less than the number of correct nodes. Prevention of faulty nodes from sending back information to the base station also leads to an improvement in the life of the network as the faulty nodes would not only

consume energy for communication but also entail energy consumption in all intermediate nodes when transmitting information back to the base station. The protocol is designed to enable self-correction in the network. The faulty nodes are isolated and they go to sleep, thereby increasing the concentration of nodes having correct information and improving their ability to generate consensus.

In the case of a large sensor network, even though the concentration of the nodes is high and the nodes cover a large region, the events being sensed by the sensors are generally localized. The wireless sensors have a fixed range and receive sensitivity. The MICA motes for instance have a receive sensitivity of –98dbm and an outdoor range of 500 ft [42]. This restricts the sensing ability of the node to monitor and sense the events outside its sensing range. The nodes would therefore be able to generate consensus only for the local information present at the nodes. The sensing and radio ranges of various nodes overlap over a particular region where the event is generated. These nodes then generate a consensus for this information in order to send it back to the base station. This decreases the energy consumption of the sensor network. The nodes in a wireless sensor network typically have the same processor performance, multi-channel radio, and electromechanical properties. The protocol has therefore been designed for a homogeneous wireless sensor network.

Different groups need to be formed when the nodes are spread over a vast area. Group formation is also required when the nodes lying in the same area have different areas of interest [4] based on their orientation. In the case of a big event being witnessed by multiple groups, consensus generated in different groups can be combined in a hierarchical manner. Different quorum systems have been proposed for providing a solution to these problems [7,8,9,11].

The whole system is divided into different subsets or groups based on their location. This is done because two nodes far from each other will not have any common information. Moreover, groups consisting of nodes close to each other would have lower communication costs in generating consensus for a given problem. The groups can be formed using various location discovery methods [2]. Various methods like radio signal strength indication (RSSI) [27] can be used to obtain approximate neighborhood information. RSSI, however, has a high error rate of up to 50% of measured distance. A better method, as suggested in [2], is that the reference nodes should have their locations predefined or estimated using GPS receivers. After the nodes estimate the distances, a distributed process of iterative multilateration starts, where each node that estimates its location becomes a reference point for others. Another method that can be used to form groups as described above is directed diffusion. Directed Diffusion determines nodes with similar interests and groups them together.

Wireless sensors can operate in either a synchronous or an asynchronous manner. The sending and receiving processes synchronize every message in a synchronous system [38]. The sending process works independently of the receiving process in an asynchronous system [38]. The asynchronous system has lower communication costs and operates in a non-blocking manner. It would thus be desirable to operate wireless sensor networks in an asynchronous system. The information gathering and consensus generating process, however, requires a limited amount of synchronization so that the information generated at a particular node has a given timestamp and consensus is not generated for an outdated piece of information. Elson and Estrin [15] present a scheme for providing limited synchronization for wireless sensors. This can be employed to give each sensor a notion of time and associate timestamps with the information present at the node. Unlike replicated servers [26], however, each node has only one piece of information and no log is kept for old information.

The energy requirements of various routing techniques are an important deciding factor in choosing a particular technique. Suppose there are p intermediate nodes from the region to the base station, the event is witnessed by m nodes in the group, and each message communication requires q micro-joules of energy. A multihop method entails transmission of information from each node to the base station and would require $p*m*q$ micro-joules of energy.

A global aggregate can be computed from the information obtained from various nodes. For a given set of nodes, a global aggregate function is computed as $f(v_1, v_2 \ldots v_n)$, where $v_1, v_2 \ldots v_n$ are the individual votes of the group members. The global aggregate functions can be computed at the cluster heads in case of a partially centralized approach or at each individual node in the case of a distributed solution to the problem. The amount of information to be sent to the base station can be reduced if a node can generate consensus on a particular piece of information. This would lead to a decrease in the number of nodes sending redundant information to the base station.

A cluster-based approach consumes less energy than the direct transmission approach. In the case of a cluster-based approach all nodes send their information to a cluster head, which in turn sends information back to the base station. The LEACH protocol [4] utilizes cluster heads for data aggregation and information gathering. Although the LEACH protocol reduces the communication costs as compared to the centralized approach, it has various shortcomings. It leads to a faster drainage of energy from the cluster head and its neighbors. Selection of a new cluster head is also an energy consuming process. If we ignore the energy required for the formation of the cluster, the cluster head reduces the amount of communication from $p*m*q$ to $(p+m)*q$. This leads to enormous energy savings for a large network. Distributed approaches therefore have less communication costs and consume less energy compared to

centralized approaches. We propose a fully distributed model for the generation of consensus.

CHAPTER 2

WIRELESS SENSOR CHARACTERISTICS

Current research in wireless sensor technology is geared toward development of a hardware and software platform that combines sensing, communication, and computing into a complete architecture [43]. Crossbow Technology is spearheading one such open source development in collaboration with researchers at University of California Berkeley. The first commercial

generation wireless sensor developed by them was called Rene Mote. Subsequent research in sensor fields has led to significant improvements in the characteristics of the wireless sensors. They have come out with the next generation wireless sensors, which are called MICA sensors. The basic MICA hardware available commercially is a square inch in size and consumes a fraction of a watt of power. Crossbow Technology is commercially distributing these wireless sensors.

Sensor nodes are fitted with an on-board processor. They have low processor and bus clock speeds, low sizes of RAM and Flash memory, and meager wireless bandwidths [16]. The hardware design of a MICA sensor mote consists of a small, low-power radio and processor board (known as a mote processor/ radio, or *MPR*, board) and one or more sensor boards (known as a mote sensor, or *MTS*, board). The combination of the two types of boards forms a networkable wireless sensor [43].

The processor being employed by the sensor is from the ATMEL ATMEGA processor family with processor speeds of up to eight megahertz. Other processors can also be used in case they meet the requirements. The processor has 128 KB of flash memory and 4 KB of SRAM. In a given network, thousands of sensors could be continuously reporting data, creating a heavy data flow. Thus, the overall system is memory constrained, but this characteristic is a common design challenge in any wireless sensor network [43].

The MPR consists of a radio as its major component, which is used to communicate with other sensors as well as the base station. The radio for MICA consists of a 916 MHz ISM band transceiver, antenna, and collection of discrete components to configure the physical layer characteristics, such as signal strength and sensitivity. The radio operates at a data speed of 50 Kbps and operates in an ON/OFF key mode. The radio can operate in either transmit, receive, or power-off mode. The absence of buffering in the radio entails each bit to be serviced by the processor in time [43].

The on-board processor allows processing of raw data, which helps in reducing the amount of data to be communicated to the base station. It is observed that the energy costs for communication are much higher than computation costs [2]. It is therefore desirable that sufficient computation is performed at individual nodes before transmitting the data to the base station. This would also lead to a decrease in the amount of data being sent to the sink.

Wireless sensor devices have very low processing, memory, and communication capacities. Wireless sensors thus make frequent transitions from active to sleep state in order to conserve energy. The nodes in the active state perform the task of receiving messages from neighboring nodes as well as observing a phenomenon. The sensor nodes move to the sleep state when there is no work to be performed in order to conserve energy. The sensors in sleep

state move into active state upon occurrence of an event. The MICA sensor motes have three sleep modes. The idle mode just shuts off the processor. The power down mode on the other hand shuts down everything except the watchdog. The power save mode is similar to power down except that it doesn't stop the asynchronous timer [42,43].

The software for the motes is developed on the TinyOS platform. TinyOS is an efficient and modular embedded software platform for the Motes. It is a component-based runtime environment designed to provide support for deeply embedded systems that require intensive concurrent operations while being constrained by minimal hardware resources [44]. TinyOS has been developed on an open source software platform by UC Berkeley with active support from a large community of users [43].

CHAPTER 3

PROBLEM STATEMENT

The events observed by the wireless sensors are generally localized. A wireless sensor can observe events in its vicinity. The radio range of the device is determined by the amount of energy used, size and shape of the antenna, the amount of interference, etc. Each wireless sensor has an area of interest, which

is determined by the sensing region of the wireless sensor. The wireless sensor observes events in the desired area of interest. Wireless sensors are deployed in a dynamic environment. A change of interest at a node would lead to different information being generated by that node. A change of interest is initiated by the user at the base station and is propagated to the wireless sensor nodes.

In order to generate consensus on the information obtained from the current area, each sensor node would identify other nodes having similar areas of interest and then correspondingly share information. We make an assumption that the sensing area of the node is no more than the radio transmission range. For instance, in case of light sensors, a sensor can only detect the light falling directly on it. The sensitivity of the node decreases as we move away from the node. The nodes being densely deployed in a region, there is no fixed line of demarcation between the visible area of adjacent nodes that would enable us to easily partition the nodes into different groups or grids. A better approach is to enable nodes to form groups based on their areas of interest and compute a local aggregate based on the local information from these nodes.

As wireless sensors are deployed in a harsh terrain and operate as an autonomous network for a long period of time, our protocol should be designed to handle process and communication omission failures as well as timing failures.

Consider sensor nodes being deployed for military surveillance to observe movements in a particular area. The nodes used to generate consensus regarding a particular piece of information would thus generate consensus on the presence or movement in a particular area. The result is then propagated to the base station. The generation of consensus would decrease the number of nodes sending back information to the base station. The improvement in the system performance is linear as it reduces the communication costs from $\Theta(k)$ to $\Theta(1)$, where k is the number of nodes witnessing the event. The protocol is therefore scalable to large networks.

A node is considered to have correct information if a majority of nodes having the same area of interest have the same value. We require that the following properties be satisfied in order to provide a solution for successful generation of consensus [38]:

Termination: Eventually each correct process sets its own decision variable and the result is sent to the base station.

Agreement: The decision value of all correct processes is the same: if p_i and p_j are correct and have entered the decided states d_i and d_j, then $d_i = d_j$ *(i,j = 1,2,.. N)*.

Validity: If the correct processes all propose the same value, then any correct process in the decided state has chosen that value.

CHAPTER 4

RELATED WORK

Various approaches have been used to generate consensus in multiprocessors and in a distributed system environment. Mirando [28] and Gehani [29] have done extensive research for fail stop processors. The basic

protocol for generating consensus by Gehani [29] uses one-to-one communication between different processors. The processor generating a consensus accesses data of other processors in a sequential order. Mirando [28] has optimized this protocol by introducing a coordinator and broadcasting messages to the various processors. The protocol, however, requires a lot of state information to be stored at different processors and thus needs to be modified so that it can be applied to wireless sensors for generating consensus. The paper provides an efficient way of implementing uniform, atomic, and causal broadcast. Wireless sensor networks on the other hand consist of wireless sensors, which behave as autonomous units and thus entail weaker consistency models like FIFO consistency.

Indranil Gupta and Ken Birman [13, 24] have devised a protocol that provides scalable and fault tolerant solutions for calculating global aggregate functions in large process groups communicating over unreliable networks. It divides the whole network into different grids, and high-level coordination is performed by protocols that aggregate the individual group member's measurements, or votes, into properties. The formation and maintenance of grids, however, requires a lot of energy and also necessitates additional state information to be maintained by the nodes in the network.

It employs a Gossip protocol [26] for exchanging information between randomly selected nodes in a group. The protocol is efficient for computing a

global aggregate for a large network that is detecting an event that is spread over a large area. We propose a fault tolerant protocol having linear message complexity for generating consensus using a variant of quorum protocols for data aggregation for localized events in sensor networks.

Various data gathering and routing protocols like LEACH [36] and PEGASIS [25] have been proposed for wireless sensor networks. The LEACH protocol, proposed by Rabiner [36], requires periodic transfer of state information in order to rotate the cluster heads and communicate this information to all the neighboring nodes of that cluster head. The PEGASIS protocol proposed by Lindsey [25] requires each node to communicate with its closest neighbor and fuse data. The nodes then take turns in sending the data back to the base station. Both PEGASIS and LEACH require communication of state information in order to maintain their topology. Our protocol on the other hand is designed to operate in a fully distributed manner. It does not require maintenance of any given topology and any node can initiate consensus (localized data aggregation) based on a random function. The random function gives higher probability to a node near the event source to initiate a consensus. Proper selection of a random function enables the protocol to perform data aggregation in a highly energy efficient manner with negligible transfer of state information. A faulty node in the chain in PEAGASIS would lead to breaking up of the chain leading to possible loss of information. Similarly the LEACH protocol would require detection of fault with the cluster head and re-election of new cluster head. Our protocol on the

other hand is highly fault tolerant compared to other protocols and can tolerate at most *(n –1)/2* faults for *n* neighboring nodes. Timeouts are used to determine failure of the node that initiated consensus. In that case again any node would initiate consensus based on the random function. Our protocol can accommodate the loss of the node initiating a consensus without any additional communication of state information. The simplicity of our protocol thus makes it highly efficient and fault tolerant compared to other protocols.

Even in the case of a large wireless sensor network, the events are usually localized. Consider a military application for wireless sensors where sensors are deployed in order to detect intrusion. These sensors can be deployed around a building or on the border. The event detected by the sensors in this case will be local. We would therefore like to perform local data aggregation and send the results to the base station. The correctness of the data can be specified by the strength (number of nodes detecting the event). It would thus be a better idea to perform data aggregation where the event is detected and then use a hierarchical model of combining data obtained from different groups thereby increasing the confidence in the sensor readings. This would further reduce the communication between the nodes and the base station. Quorum consensus algorithms can be employed for reaching consensus because their cost is considerably less than that of other algorithms used in Gossip [26] and Bayou [45,46] systems. The Quorum Consensus (QC) algorithms follow the method of determining and eliminating the malicious node

from the group or introducing a sufficient amount of redundancy in the group to accommodate these errors.

Quorum consensus methods are used in distributed systems to generate consensus for a particular set of servers or processors in a particular network partition. Gifford [3] developed a file replication scheme using weighted votes for each particular replica. A read operation takes place by obtaining a Read Quorum. Quorums can also be extended to sensor networks. For large networks we can combine different quorums to form a large quorum system. Various quorum systems [7,8,9,11,22] have been proposed in order to resolve different problems and increase the availability and efficiency of the replication services. In the case of wireless sensor networks the sensing range of a network can safely be assumed to be less than the radio range of a sensor. We can therefore generate a read quorum for the information in a particular area. We are, however, not using Maekawa's algorithm [11] for determining the quorum size. The quorum size is determined by the group size, whose size is in turn determined by the sensitivity range of nodes in a particular area.

The primary purpose of our protocol is to compute a local aggregate and send back this information to the base station in an energy efficient manner. Various epidemic quorum systems have also been proposed for data replication. For instance Keleher [21] presents a protocol for data replication by combining epidemic information propagation with voting in weakly connected and mobile

environments. We can perform data aggregation at a few nodes in the group and then send this information back to the base station from only these nodes in the group. We propose a variant of Quorum protocols for obtaining local aggregate in the sensor network.

Synchronous networks consume more energy as compared to asynchronous networks. As energy conservation is very important for a wireless sensor network, it can be designed to work as an asynchronous network [15,16]. The classical result of Fischer, Lynch, and Patterson [14], however, states that it is impossible to generate consensus in an asynchronous environment with even one faulty process. Introduction of synchronization in a system facilitates detection of faults by using timing failures (timeouts). Moreover since the sensor data is generated from real (as opposed to logical) phenomena; there is loose synchronization inherent in the system. It would thus be desirable for wireless sensor networks to operate as a synchronous network in our case. Limited synchronization can be provided for short intervals by using an external clock [15]. This synchronization is done at the beginning of the consensus generation process. A quorum is used to enable a network system or group with n nodes to tolerate at most $(n-1)/2$ faults.

CHAPTER 5

PROTOCOL REQUIREMENTS

We have devised a new protocol for generating consensus in wireless sensor networks. A wireless sensor network is an autonomous system and is designed to operate in harsh terrain for long periods of time. A wireless sensor

network protocol should thus be designed to be resilient to faults in order to minimize supervision and maintenance costs. The protocol needs to be scalable in terms of computational overhead and power consumption at sensor nodes. The state information or the packet load should also not increase exponentially with an increase in the number of nodes. Two different architectures can be used for wireless sensor networks. A centralized architecture has higher latency compared to a distributed architecture. The nodes in the centralized architecture may be saturated by data aggregation requests from multiple nodes for different information generated in different areas of interest at various nodes. The centralized approach is less tolerant to faults as loss of information will occur when the central node goes down. A distributed architecture would thus scale better and be more fault tolerant than a centralized architecture.

As most events detected by a wireless sensor network are local to a particular area, we propose a new fully distributed protocol for data aggregation in a local region. The fully distributed consensus generation approach is more efficient and provides better fault tolerance. It makes the wireless sensor network operate as a self-correcting network by isolating faulty nodes in the network and making them go to sleep. The nodes go to sleep when the probability of the node to generate consensus becomes less than the threshold probability value required for generating consensus.

Wireless sensor nodes are resource-constrained devices. A completely

decentralized approach toward data aggregation also increases the life of the network. Each node, on detecting an event, waits for a random amount of time before initiating consensus. The introduction of a random wait time not only distributes the process of consensus initiation to different nodes in the network but also reduces the number of nodes generating consensus. It also leads to a uniform dissipation of energy in the network, as different nodes initiate consensus for different information.

A random node in the local region starts the process of generating agreement for its own information. This process suppresses the process of consensus generation in nodes having the same area of interest. The nodes that have the same information as the node initiating the consensus send back a positive acknowledgment to the node initiating consensus. The nodes having different information send a negative acknowledgment to the initiating node and wait for the result of consensus. The result of consensus is replicated in nodes having the same area of interest. In case the nodes do not receive any response to the consensus request and the process times out, a node would then initiate consensus after a random amount of time. A node starts this initiation process only after determining that its information is new and no consensus has been generated for this information.

The fully distributed system decreases the amount of state information that must be kept at a particular node. The nodes do not need to store the

information about the leader of the cluster. They also do not need to store information about nodes in the network other than their neighbors.

The amount of communication among the nodes is also decreased, as the nodes do not need to elect a leader. All wireless sensors operate as autonomous units. They work as independent entities and communicate only to generate consensus amongst the nodes. A node generates consensus for its information and therefore does not need periodic transfer of state information. This holds true for a mobile network as well. As the nodes behave as autonomous entities, they do not require causal or total ordering coordination among them. Even though consensus requests initiated by different nodes are mutually exclusive, consensus requests from the same node might overlap in extreme circumstances. The protocol can, however, easily be extended to provide FIFO consistency by attaching serial numbers with the messages in order to identify them with the corresponding consensus request.

We believe that our study is the first one that has proposed the use of quorum techniques for solution to the problem of generating consensus in wireless sensor networks. The quorum approach has traditionally been used for data replication in different databases as well as in multiprocessor environments [3]. The wireless sensor network is an ad-hoc network that primarily has low power consumption and a limited amount of memory. It is not possible for these sensors to store multiple values or maintain a log of values based on their

timestamp. Our approach minimizes the amount of state information required at each sensor node. It also minimizes the amount of communication taking place for maintenance of state information.

CHAPTER 6

CONSENSUS PROTOCOL

6.1 PROTOCOL DESCRIPTION:

A consensus is required to generate agreement about particular information present at different servers or nodes. We have devised a new protocol for generation of consensus in wireless sensor network. It uses a variant of quorum techniques for efficiently generating consensus and tolerating faults in the system. In our case a quorum is a subgroup of wireless sensor nodes whose size gives it the right to carry out operations. For a sensor network consisting of *n* nodes, and *k* nodes having same area of interest, the size of the subgroup (quorum) should at least be at most *(k + 1)/2*. The process for consensus generation is described below.

A large concentration of nodes exists in a small area. Each node has a certain sensing region *S*. A node obtains a sensor reading when a user generates interest in an event, and that request is propagated to the sensors in that region. The sensors are activated and collect information in their respective regions. The sensors then initiate a consensus generation process. In order to generate consensus on a particular sensor reading, there should be a sufficient number of nodes to achieve a quorum. The nodes within the sensing range of a particular sensor may have different readings from the information for which the consensus is being generated. This can be attributed to various reasons such as malfunction of some nodes or nodes generating consensus having a different area of interest or even change in the sensing region *S* of a node over a period of time. The sensor readings obtained by a node may not therefore conform to the sensor data generated by other nodes in the same area. Thus we observe that

though the node may not be faulty, it could have a negative impact on the decision process when the sensing region of the node is different.

Consider a network consisting of n nodes. Each node, n_i, maintains a state variable, p_i that measures the probability of this node to generate consensus regarding its information. The nodes generate consensus by obtaining a Read Quorum from nodes having similar interests and area of coverage. The value of p_i is increased if the information at the i^{th} node is the same as that for which the consensus has been initiated. On the other hand the value of p_i is decreased when the information at a node is different from the result of the consensus.

The initial value of p_i is selected based on the redundancy in the network and the accuracy required in the network. With an increase in the redundancy of a group or region, the effect of a node going to sleep on the ability of the group to generate consensus reduces. The value of p_i would thus depend on the number of nodes in the group. Suppose the group g has C_g nodes, then the initial probability p_i of the node to generate consensus is

$$p_i = K/C_g \qquad (3)$$

where K is a constant.

A quorum is obtained after nodes exchange information amongst themselves. We can call this exchange of messages between different nodes a consensus generation process. A consensus generation process is initiated by a

node having information for which consensus has not been generated. Other nodes that are part of the group of the node initiating consensus relay back a positive or negative acknowledgment based on whether the information present at the nodes conforms to or differs from the information of the node initiating the consensus. The node initiating a consensus then maintains a count of the acknowledgments received. For a group of n processes or nodes, the total number of positive acknowledgments (including itself) is greater than or equal to $(n+1)/2$ signifying that the node has been able to generate consensus for its own information. We call this procedure a Read Quorum.

A node, before initiating a consensus for its own sensor readings, would compare the value of p_i with the threshold probability P. The process can be initiated only if $p_i > P$. When the value of p_i becomes less than P, it implies that the node is unable to generate consensus consistently. The node thus needs to disband the old group and form a new group with similar interests. Another factor that should be taken into account before consensus initiation is the energy level at the node. The node should have sufficient power for transmission and reception of signals.

In case the probability p_i of node drops below P the node would be unable to either initiate consensus or participate in a consensus process. It then moves to the phase of forming a new group that has similar interests. When the node is unable to form a new group, the node can be assumed to be faulty and goes to

sleep. The redundancy of the network enables the node to have a sufficient amount of time to form a group before it stops trying and goes to sleep. This ensures that no node is incorrectly put to sleep. The system has sufficient redundancy to accommodate the elimination of faulty nodes.

A wireless sensor network is an autonomous network. The network should thus monitor itself by identifying and correcting faults in the system. The advantage of self-correction in wireless sensors is enormous. It reduces the amount of state information kept by each node. This leads to a reduction in computation and communication costs as well as improves the operational life of the network.

In our design each node is required to maintain only a small set of state variables (the group identifier, a list of members in the group, or the number of nodes in the group, and the probability variable p_i). It also reduces the communication costs significantly as the nodes need to transmit only a small amount of state information. As each node operates independently, it makes its own decision about the node itself being faulty. This leads to a considerable savings in energy, as the cost of communication between nodes is relatively high. The computation at each node is also decreased as the node computes its own probability p_i only. The node makes decisions based on its own probability values. Moreover the nodes do not need to inform other nodes about faulty or correct nodes.

A situation can arise where the node initiating a consensus can experience faults and might be unable to generate consensus. The protocol has been designed so that other nodes in the vicinity experience timeouts and initiate consensus after a random amount of time. This enables the system to tolerate faults in the node initiating a consensus, besides the $(n-1)/2$ faults for n participants.

A node initiates consensus after a random amount of time based on a random function. The random function should be chosen carefully in order to prevent concurrent initiation of consensus for the same information along with a minimal increase in latency. The sensitivity of a sensor decreases with an increase in the distance of the event from the node. A node closer to the event should have a higher probability of initiating consensus for the event, as there is a higher probability that the neighboring nodes would have witnessed the event. Wireless sensors are generally employed to detect light, heat, sound, or other phenomena. Wireless sensors can employ signal strength to determine the distance of a source from the wireless sensor. For instance, in the case of image sensors, the light collected by the sensor aperture is inversely proportional to distance of source squared. Carlson [19] provides a method for measuring the signal-to-noise ratio (SNR) and relating it to pixel sensitivity for an image sensor. He employs the following relation for collection of light by the sensors:

$$E_{fp} \propto 1/D^2, \qquad (4).$$

where E_{fp} is the light collected by the optical system (focal plane illuminance) and D is the distance of the light source from the sensor.

We have chosen the random function to be modulo k times the number of neighboring nodes where k is a constant that can be varied to obtain a tradeoff between latency and energy consumption in generation of consensus. The random function R would thus be

$$R = (rand\ () / D_0)\ mod\ (k * N_i), \qquad (5)$$

where N_i is the number of neighboring nodes, D_0 is the distance of the object or event source from the node, and *rand ()* is the random number generated by the node.

In the case of thermal sensors and certain other sensors where the distance from the source cannot be determined, the random function can be defined as.

$$R = rand\ () * |T - t_i|\ mod\ (k * N_i), \qquad (6)$$

where T is the average intensity and t_i is the intensity at the sensor node. The intensity can be either signal strength or temperature or light intensity. The initial value of T can be chosen at the time of deployment and the value is then changed at each node based on the values of t_i. The sensor node keeps a count c of the number of values of t_i and computes the new value of T as follows

$$T = (T * (c-1) + t_i) / c \qquad (7)$$

We thus observe that a fully distributed design of the sensor network not only leads to a simplification of the whole process but also a significant increase in the efficiency of the system.

6.2 PROTOCOL DETAILS:

Our protocol delegates the task of generating consensus to individual nodes as compared to the multihop networks, where the cluster head or a coordinator handles the task of generating consensus. Each node maintains a list of neighboring nodes (or a count of the number of nodes) having similar interests for generating consensus. The nodes send out control signals in the form of beacons in order to inform other nodes about a change of interest or other control information like the node going to sleep. This push architecture is more conducive for our environment as it reduces energy consumption by the nodes of the system.

Assumption I**:** The nodes are assumed to be homogeneous. All the nodes have the same computation and communication power.

Assumption II: The sensing range of the wireless sensor is less than the radio range.

The control messages are broadcast to the neighboring nodes of the system that are within the radio range of the node. A control message is

transmitted using a fixed structure. The various fields used in the data transmission structure also constitute the state information present at the node and are shown below:

Src	*CType*	*Area of Interest (Optional)*	*SData*

Src: The wireless sensor identifier

CType: The type of message

SData: The data payload

Area of Interest: Area of Interest. This field is optional. The Area of Interest is used only when the nodes store CVCount instead of list of members in the group. The tradeoff is between communication and storage costs. It is desirable to store a list of members in case of a static network and area of interest in case of a mobile network.

A node communicates with neighboring nodes in order to reach a consensus on certain information. Each node also maintains additional state information, which is used for the consensus generation process. A node should contain a set of data structures for storing information about other nodes having similar information. A connection vector, CV, as specified by Pardoe [6] can be

employed for storing a list of nodes having common interests and within the radio range of the wireless sensor. All the nodes keep their own CV. We can have different variations for the implementation of the protocol with the CV. Each node can store the CV of the corresponding nodes in the group. The CV can represent each node by a bit [6] or an integer value. It would then send messages to the nodes in its CV and receive their replies. The other variation can be that the nodes in the group can store only the group identifier and the number of nodes in the group represented by *CVCount*. All information is broadcast to the nodes in the group. This minimizes the amount of state information to be stored in the nodes.

A state variable, CVCount, stores the number of nodes in the group *i.e.*, the nodes having the same information. Another variable p (p_i) stores the probability of the node to generate consensus. The value of p is initialized based on the value of CVCount at the node. In other words,

$$p = K/ CVCount$$

The control messages should contain the following flags (*CType*). The control messages are broadcast to all the nodes within the radio range of the node.

CType – type of message.

INIT

GETINIT

CONSENSUSINITIATE

INITCONSENSUS

SLEEP

AWAKE

DOWN

INTERESTCHANGE

CONSENSUSRESPONSE

CONSENSUSRESULT

The control messages are described below:

INIT – The *INIT* message is sent by the node to advertise its interest. The receiving nodes add the node to their CV or increment the CVCount (based on the nature of implementation) if they belong to the same group and have the same interest.

GETINIT – This message is sent by a node in order to request *INIT* messages from the nodes that are within the radio range of the sensor node. A node sends out this message when it wakes up and requires initialization of the state information.

CONSENSUSINITIATE – This is an internal control message. It begins the process of consensus initiation by a node in the network. The node, upon obtaining new information, uses this control signal internally to initiate the consensus generation process by the node. The node initiates consensus by broadcasting its information with the *INITCONSENSUS* control signal.

INITCONSENSUS – Nodes in the group receive the *INITCONSENSUS* signal when a node initiates consensus in the group. Based on the information present at the node, the node examines whether its information matches the information received with the message.

CONSENSUSRESPONSE – The reply to the *INITCONSENSUS* message is sent back by using the *CONSENSUSRESPONSE* signal. This signal informs the node that initiated generation of consensus about the nature of information of the nodes replying with the *CONSENSUSRESPONSE* message. An affirmation is sent in case of a match. In case of no match, the node sends a negative reply to the source.

The affirmation or negation is specified in the data header of the control message carrying the *CONSENSUSRESPONSE* message. The format used for the reply is specified below:

Decision	*ID of node initiating the consensus*	"-"	*ID of the node responding to the message*	"-"	Source Data

The Decision field holds the value "Y" if the information at the node is the same as the information received. Else it has an "*N*". The two node identifiers are separated by using a "-".

For instance, if the node with ID 12 initiates a consensus and the node with ID 30 having the same information receives the consensus initiation request, it will reply with an affirmative signal having the following format:

Y12-30

SLEEP – This message informs the other nodes that the sending node has gone to sleep. This message is broadcast to all the nodes in the group.

AWAKE – This message specifies that the node has started operating actively. The node subsequently sends out the *GETINIT* signal to garner information required to initialize the node.

DOWN – A node sends out this message when this node experiences a malfunction and wishes to stop performing normal activity.

INTERESTCHANGE – This control message is sent out when there is a change of interest at the node. The old interest is sent out along with the new interest, as the neighboring nodes will update their CV or CVCount based on a comparison of their areas of interest with the old area of interest. In case the old area of interest matches with that of the node, then CVCount in decremented or the node is removed from the CV based on the implementation. On the other hand, if the new area of interest matches the area of interest of the node then either the CVCount is incremented by one or the member is added to the CV based on the implementation. The structure of the *INTERESTCHANGE* message is:

Src	*CType* *INTERESTCHANGE*	*Old* *Interest*	*New* *Interest*

CONSENSUSRESULT – This control message is sent out with the result of consensus to all the nodes in the neighborhood. The nodes that had participated in this consensus generation process will then accept this information.

In case the consensus-initiating node received an affirmative response, it will determine whether it has achieved a sufficient number of yes votes to obtain consensus. Otherwise the number of negative votes is incremented by one.

The total number of votes is then examined to determine if consensus can at all be generated. The value of p_i is incremented in case a consensus is generated and decremented if the node is unable to generate consensus. If the value of p_i becomes less than a given threshold it sends out a GETINIT message after some amount of time and examines the reply messages. In case its information is the same as that of the majority of nodes with the same area of interest, it will form a new group, else it will go to sleep. We are thus able to generate consensus in a localized distributed environment and also accommodate limited amounts of faults.

Each node requires a list or count of nodes having a similar interest in order to generate a consensus. The area of interest is dependent on the

sensitivity of the sensor. The nodes send out an *INIT* message during the initialization phase in order to inform nodes having similar areas of interest and enable each to generate their own CV for future purposes. The CV contains information like id, location, etc. of different nodes. In order to minimize the state information being stored by the node, the nodes can just keep CVCount instead of CV. CVCount stores the number of nodes having similar interests and lying in the neighborhood of the node.

The node modifies its CVCount when it receives one of the control messages *AWAKE*, *SLEEP*, or *DOWN*. The CVCount is decremented upon receiving the *SLEEP* or *DOWN* control packet. On the other hand, the CVCount is incremented upon receiving an *AWAKE* signal.

The geographical coordinates of the area where the node is active usually specify the interest of a node. We can reduce the amount of communication between different nodes by transmitting control information only when there is a change of interest at the node. The change of interest at a node n_i would entail the node sending out control message beacons to the neighboring nodes. The advertisement messages will have a dual effect on the neighboring nodes. The nodes that have n_i in their CV will decide on the further status of node n_i. The nodes might still keep n_i in their CV when there is not a large change of interest in the nodes. Other nodes that did not have n_i in their CV will make the decision

of including the node n_i in their CV. CVCount will be altered accordingly if it is used instead of CV.

Concurrent initiation of consensus should be prevented at multiple nodes for the same information. This is done in order to reduce the message complexity from O (N^2) to O (N) in the ideal case. The nodes send a reply to the initiating node and stop generation of consensus at their node. On receiving the result of the consensus they mark their information as old and replicate the result of the consensus. A timeout will occur when the node that initiated consensus encounters a fault. In that case another node initiates consensus for its information after a random amount of time. This ensures that consensus is generated in case the source initiating a consensus encounters faults.

The system is designed to be self-correcting in nature. The nodes work as independent entities and decide themselves whether they are faulty. Each node has a probability of generating correct information. The information is assumed to be correct if a quorum can be obtained for that information. The probability is decreased in case the node is unable to generate consensus for its own information. In case the probability of the node drops below a given threshold it signifies that the node has been consistently unable to generate a consensus. The node then goes to sleep for some amount of time. This procedure of consensus generation would continue until all the nodes have initiated consensus about their information.

Concurrent initiation of consensus is prevented by the use of timestamps as the sensors have a notion of time due to limited synchronization provided by an external clock. The sensor node that initiated the consensus first is the node that finally generates consensus. Consensus is stopped at all the other nodes. Due to a high density of nodes in the network neighborhood, the propagation delay can be assumed to be negligible.

The protocol design enables the dynamic addition of nodes to the system. A node joining the network will send a *GETINIT* control message in order to initialize its state information. It will also send out an *INIT* message to the other nodes in its radio range in order to inform them about its interests.

A probability for generating consensus is associated with each node. The probability decreases when the node is unable to generate consensus for its information. The probability of the node to generate consensus is increased when it successfully generates consensus.

The node will receive correct information when the group is initially formed. Each subsequent generation of consensus after the first one will have a different probability value. The node goes to sleep when the probability of the variable becomes less than the defined threshold value for the system.

The node wakes up after a random amount of time. The node, upon waking up, broadcasts a *GETINIT* message to form a group of nodes having similar interests and to obtain the CVCount. It also broadcasts the *INIT* message to other nodes to inform them about its own interests. The node then behaves as any other node and tries to generate consensus for its own information.

The above procedure shows that the node itself decides group membership. This helps to decrease the amount of status information kept by various nodes. It also ensures greater security in the group. In case a particular node starts malfunctioning or is overtaken by an enemy, the node will be unable to generate consensus about its information and will subsequently go to sleep. Thus the system behaves as a self-correcting network and makes the network robust.

6.3 Proof

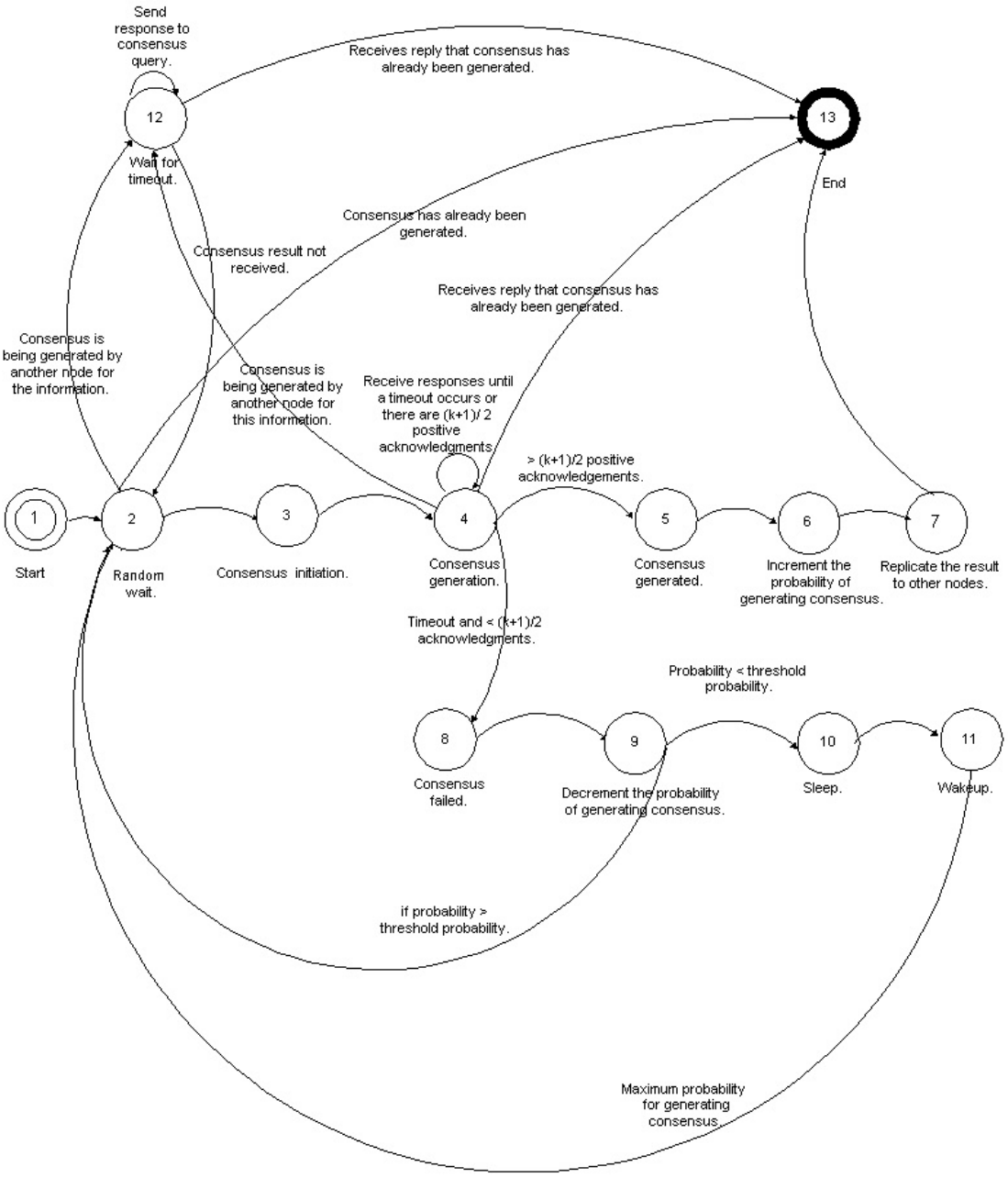

Figure V: Protocol Description Using A Finite State Diagram

Lemma 1: A node is able to successfully generate consensus for its information when there are no faults in the system.

Proof: In the absence of faults in the system, all nodes will observe correct behavior and will thus have the same information. Assuming there are *k* neighboring nodes. Consensus is initiated by one of the nodes in the neighborhood (state three) and a request is sent to the other *k-1* nodes in the neighborhood. It will then receive correct responses from the participating nodes (state four) and evaluate the values by using a simple majority to obtain the correct result (state five). The node that generated a consensus sends the result (state seven) to other nodes and goes to the end state.

The termination property is satisfied in this case by two things. A majority is always achieved by nodes that observe correct behavior. The node that generates a consensus propagates the result (state seven) successfully to other nodes (states two, twelve).

Agreement and integrity are satisfied as the result of consensus is replicated at the participating nodes. All the correct nodes thus participating in the consensus operation will have the same value after the consensus operation.

All the wireless sensors in a particular area cannot have the same area of interest. The lemma still holds in that case. It is possible that not all nodes in a group observe the same event. Consensus will be generated if a majority of nodes in the group witness an event. In case, a minority of nodes in the group witnesses an event, the nodes will successively try to generate consensus and

will subsequently go to sleep (state ten). Each node forms a new group upon waking up (state eleven) based on the nodes having the same data for that particular timestamp. The new group will thus be smaller than the original group and will be able to generate consensus for the information. This ensures that consensus is initiated (state two) and eventually generated (state four) by correct nodes in a group. This can also be explained by an example. In case only three of the seven nodes are able to detect an event, these three nodes will be unable to generate consensus. Their probability to generate consensus will thus decrease and they will eventually go to sleep at different times. The nodes upon waking up, will form a new group consisting of three nodes and will thus be able to generate consensus (state four) for their information. The termination property is satisfied when the process successfully propagates the result to members of the new group.

Agreement and Integrity are satisfied as the nodes that detect a particular event successfully receive the result of consensus and thus all correct processes have the same value. The correctness or the correct behavior of a process in this case is defined by the ability of the node to detect the event within its area of interest.

Lemma 2: Consensus is initiated by at least one node in the network neighborhood.

Proof: Each node waits for a random amount of time in state two before going to state three and initiating consensus. Random wait ensures with high probability that all nodes do not initiate the consensus process at the same time. The node determines whether consensus is being generated for the information while it is present in state two before going to state three and initiating consensus for the new information. In case consensus is being generated for the information, the node goes into wait-state (state twelve) till there is a timeout. The node goes to end state (state thirteen) if it receives the result of consensus within a timeout period. In case the result is not received within the timeout period, the node goes to state two and starts the whole procedure again. This ensures that each node keeps on trying to initiate consensus in network neighborhood till it either receives the consensus result (state twelve) or initiates the consensus itself. Consensus is therefore initiated by at least one node in the network.

Lemma 3: Concurrent generation of consensus is prevented.

Proof: Timestamps are used to provide a notion of time to the various nodes. Random wait (state two) reduces the number of nodes trying to generate consensus at the same time. The consensus initiation and generation process at other nodes is stopped (state twelve- wait state) upon receiving information that another node is generating consensus for the same information. The wireless network also prevents concurrent use of the wireless channel without destructive interference. The process with the oldest timestamp is allowed to generate consensus while the other processes go to the wait state (state twelve). A

consensus initiation (state three) request will thus be sent by only one node at a particular time. The participating nodes then wait for a timeout (state twelve) and go to the end state (state thirteen) upon receiving the result of consensus or try to initiate consensus after a random amount of time. This ensures that consensus is not being generated at multiple nodes at the same time. The replication of a consensus result (state twelve) by the initiating node ends the process at participating nodes (states two and four).

Lemma 4: Each correct node gets the final result of consensus.

Proof: The participating nodes wait for the result of consensus (state twelve) till a timeout occurs. In case the nodes do not receive the result of consensus, they initiate consensus after a random amount of time (state 2). They will then get the result of consensus from neighboring nodes (receive reply in state four) if the consensus has been generated and the node has not received it due to communication omission or timing failures. If on the other hand, the earlier consensus generation process has either been unsuccessful (state eight) or the node crashed due to a process omission failure, the current node initiates the process. It will require $f+1$ rounds (or $f+1$ nodes to initiate consensus) for an upper bound of f process or communication omission failures for generating consensus for $2f+1$ nodes. The process will thus end successfully.

Lemma 5: The protocol functions correctly for up to f failures.

Proof: The protocol works correctly in case of an upper bound of f process or

communication omission failures, where $f = (n-1)/2$ for n participating nodes. The initiating node obtains $f+1$ correct and up to f incorrect values in state four. A consensus is thus achieved in state five and the result has the same value as that of $f+1$ nodes. The termination condition is satisfied when the participating nodes receive the result of the consensus as proved in *Lemma 4*. The protocol thus terminates with the result of consensus replicated at all the nodes in the network. Agreement and integrity are ensured when the nodes receive the result of the consensus and have the same value at the end of the process.

The protocol also functions correctly for the case where wireless sensors have non overlapping areas of interest as proved in lemma one.

The node initiating consensus can itself encounter faults during the process. A neighboring node will then start the process (state three) and the worst case would involve $f+1$ rounds for generation of consensus and dissemination of information to the base station and participating nodes. Agreement and integrity are ensured as all the correct processes have the same value at the end of the process.

CHAPTER 7

PROTOCOL ANALYSIS

7.1 Concurrency:

Our model enhances the ability of the system to make decisions concurrently. A multihop or a hierarchical network makes the decisions at the base station or other uniprocessor devices, which can either be a cluster head or other intermediate node. A cluster head caters to multiple nodes. A situation can arise when more than one node want to concurrently initiate consensus for different information. It can also occur that consensus for a particular piece of information may take a longer time due to process or communication omission failures or timing failures. In the meanwhile another node may want to initiate consensus on an entirely different sensor reading. The concurrent initiation of consensus by the cluster head will introduce latency in the system, as the cluster head is a uniprocessor device.

The cluster head would require additional buffer space to store the results received from different nodes for different pieces of information. It would also require additional computational power in order to manage multiple sessions simultaneously. If the number of nodes participating in the consensus is *n* and the number of concurrent consensus requests is *m*, concurrent consensus generation would entail an increase in the buffer space and computation power of the cluster head by a factor of *m*n*. The space complexity of the system would

thus be O (*m*n*), which greatly increases the complexity in the design of a sensor network.

On the other hand our model is much simpler. It requires the node to generate consensus for its own information. In case multiple nodes with different areas of interest and the same radio range want to generate consensus for their information, they can generate consensus independently without increasing latency and computation. The source node collects information from nodes with similar interests and generates consensus on its own. This makes the whole process of generating consensus at different nodes independent of each other.

The independence of the various nodes in generating consensus also effectively eliminates the need for a distributed mutual exclusion algorithm for consensus generation compared to a cluster head approach. The nodes operate independently of each other and are thus not affected by other nodes. We thereby improve the scalability of the system as well.

7.2 Consistency:

Distributed systems employ different techniques for providing synchronization. They can have external or internal synchronization. In the case of wireless sensors, the emphasis is on minimizing energy by reducing the amount of communication bandwidth required for synchronizing communication

among the nodes. Periodic transmission of beacons is an energy consuming process. As sensor nodes are energy constrained devices and the synchronization process is an energy consuming process, the possibility of internal synchronization is ruled out.

External synchronization requires an external source to send out beacons periodically. The nodes receive beacons and synchronize their local clocks accordingly. Elsin and Estrin [15] present an optimization of the process by providing limited synchronization when the nodes initiate communication. The synchronization can thus be performed before a node in the sensor network initiates a consensus process.

External synchronization enables our model to scale in the same way as a multihop or hierarchical network. The nodes in a region will be synchronized by an external clock irrespective of the topology of the network. There will be absolutely no difference in terms of energy and other factors required for providing synchronization.

The wireless sensors are memory constrained devices and can contain only one piece of information. Moreover, wireless sensors are autonomous and the information present at each node is independent of the information at the other nodes. Vector timestamps are therefore not advisable for wireless sensors. External synchronization can be used to provide timestamps for the information

present at the nodes.

7.3 Fault Tolerance:

Our protocol enhances the fault tolerance of the system. The protocol has been designed to tolerate timing and omission failures.

A process omission or communication omission failure in the localized approach does not affect other nodes. Each node has the capability of generating consensus for its sensor readings. In case a node does not initiate consensus and encounters failures, redundancy in the system will ensure that the other nodes are able to generate consensus for the sensor readings in the region. The system has sufficient redundancy to accommodate failure of nodes replying to the consensus request. For a group of *m* nodes, the sensor network can accommodate at most (*m-1)/2* omission and communication omission failures. Failure of the node initiating consensus would lead other nodes having the same area of interest to initiate consensus for their sensor readings.

The hierarchical approach is more vulnerable to failures than the distributed approach. The cluster head communicates with multiple nodes (assuming *m*) and generates consensus based on the information received from these nodes. The cluster head transmits this information to either the base station or other cluster heads. As all communication takes place through the cluster head, failure of the cluster head would lead to loss of information from the

m nodes communicating with the cluster head. Our protocol resolves the problem by enabling neighboring nodes to initiate consensus after a timeout period.

In case of consistent communication omission failures, the faulty node would be unable to consistently generate consensus and would thus eventually try to form a new group and upon subsequent failure would go to sleep. The consensus process can accommodate at most (*m –1)/2* failures for *m* participants. In case the node initiating a consensus encounters failures, other nodes in the group will time out and initiate the process after a random amount of time. This enables the system to be highly robust to timing and omission failures.

We thus observe that our approach has better fault tolerance for generating consensus than the hierarchical approach does.

7.4 Efficiency:

The efficiency of the whole system is much better than the hierarchical approach. The hierarchical approach requires different nodes to become the coordinator (cluster head) after some amount of time. The transfer to a new coordinator requires synchronization between different nodes and cluster heads along with advertisement messages being broadcast by the coordinator to signify the beginning and end of the coordinator position. This procedure is similar to a Token Ring (rotation of token takes place to enable each node to transfer data and crash of node possessing the token is equivalent to crash of cluster head)

and would thus require similar error control features as well. For instance, the system should be able to recover and choose a new coordinator in case the current coordinator encounters a crash failure.

The complexities of the hierarchical approach are effectively mitigated by our protocol, which uses local algorithms for generation of consensus.

The major performance issues for a distributed system are responsiveness, throughput, and balancing of computational loads. Our system scales better in all the three mentioned parameters. The system is more responsive to detecting changes in the local area. Each node forms its own group consisting of nodes with similar interests. This improves the probability of the node to reach a consensus for its information. The system is thus more sensitive to information being generated.

Other multihop and hierarchical approaches are less responsive compared to our model. The list of nodes that are part of the group may not have the same interests and thus would have a negative impact on generating a consensus. This could lead to a decrease in the sensitivity of the system and therefore make the system less responsive. Hierarchical systems might be unable to generate consensus for events that occur on the boundary of two clusters leading to formation of blind spots in the network. Our protocol functions correctly in these cases and thus has better sensitivity than these networks. We

also observe higher latency in hierarchical networks when the system is required to make concurrent decisions as discussed earlier. The fully distributed approach allows concurrent generation of consensus leading to a higher throughput of the system.

The load of the system gets distributed as each node is given the ability to generate consensus for its own sensor reading. Each node can generate consensus when desired and go to sleep in case of malfunction. We thus observe that our protocol has better responsiveness, throughput, and load balancing than other models for generation of consensus in wireless sensor networks.

CHAPTER 8

SIMULATION RESULTS AND DISCUSSION

The simulations were done using the TinyOs simulator built by University of California-Berkeley. The simulator uses a well-defined programming model by having a modular architecture in which different components combine to form an executable. The program is written in NESC and the simulation results are obtained by running TinyOs on a Windows 2000 machine. The protocol was simulated to obtain results under different conditions in order to verify that the protocol works in accordance with the assumptions.

The first set of results was obtained to study the variation of consensus initiation requests with the increase in the density of the network. The number of neighboring nodes in the radio range of a sensor increases with the density of the network. The random function is chosen in order to enable the nodes near the event source to have a higher probability of initiating consensus for an event. The value of the product $k*N_i$ in the random function ($R \propto 1/ D_0 \ mod \ (k* N_i)$) is selected based on the tradeoff required between latency and communication costs in the network. The following series of tests were conducted by keeping the random function as twice the number of neighboring nodes. For instance the random function is 40 for 20 neighboring nodes, 60 for 30 neighboring nodes, and 100 for 50 neighboring nodes.

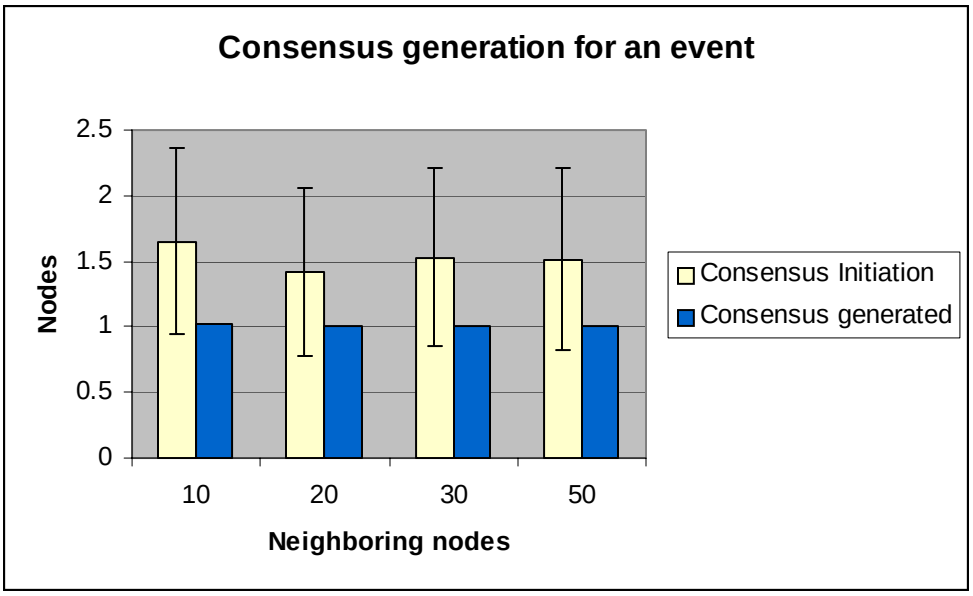

Figure VI: Variation Of Consensus Initiation For An Event With Different Node Density

It can be observed from Figure VI that the number of nodes initiating consensus does not increase with the density of the network. The redundancy of the network can thus be increased without a significant increase in the number of nodes initiating consensus in the network. The protocol is thus scalable to networks having higher node densities.

Concurrency is another major issue in distributed systems and wireless sensor networks. The second set of tests was done for generating consensus with multiple events in the sensor neighborhood. The sensitivity range of wireless sensors is less than the radio range of the sensors. Multiple events can occur within the radio range of the sensor but outside its sensitivity range as shown in

the figure below. The CVCount is thus formed based on the nodes having same sensitivity or similar interests within the given radio range.

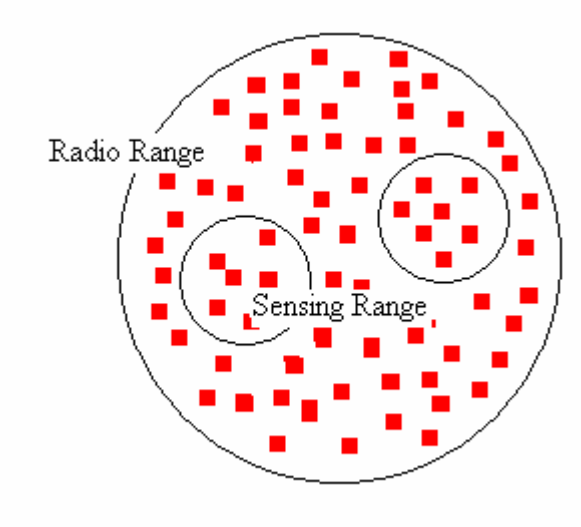

Figure VII: Sensing And Radio Range Comparison In A Sensor Network

The following results were obtained for generating consensus for multiple events in a sensor network. The protocol generated consensus successfully for multiple events without increasing the number of nodes initiating consensus as shown in Figures VIII and X. It can thus be observed that the protocol is scalable for generating consensus for multiple events in a sensor network.

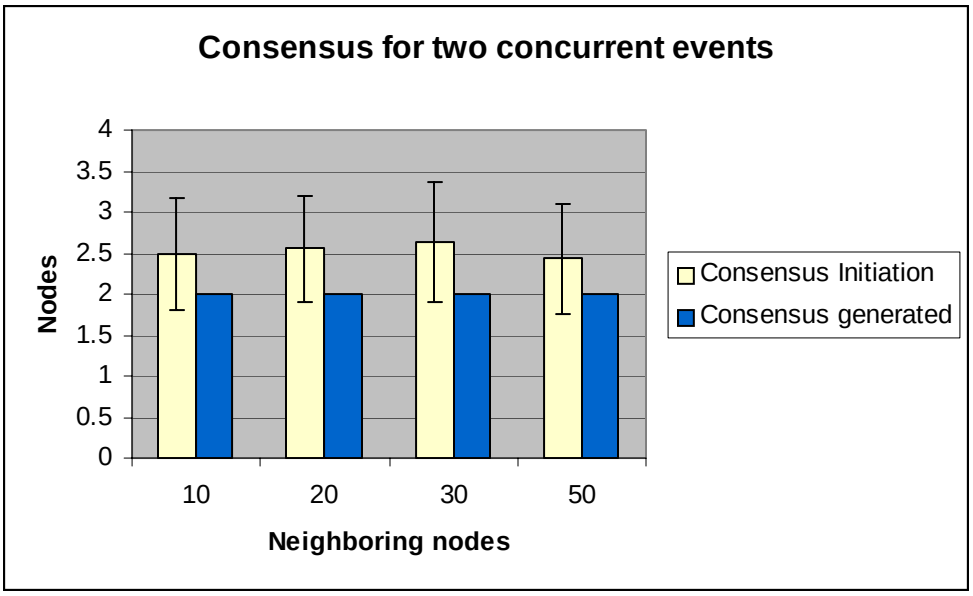

Figure VIII: Variation Of Consensus Initiation For Two Concurrent Events With Node Density

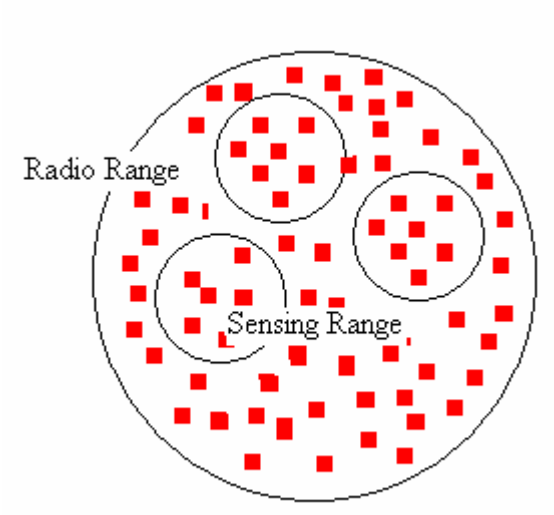

Figure IX: Sensing And Radio Range Comparison In A Sensor Network

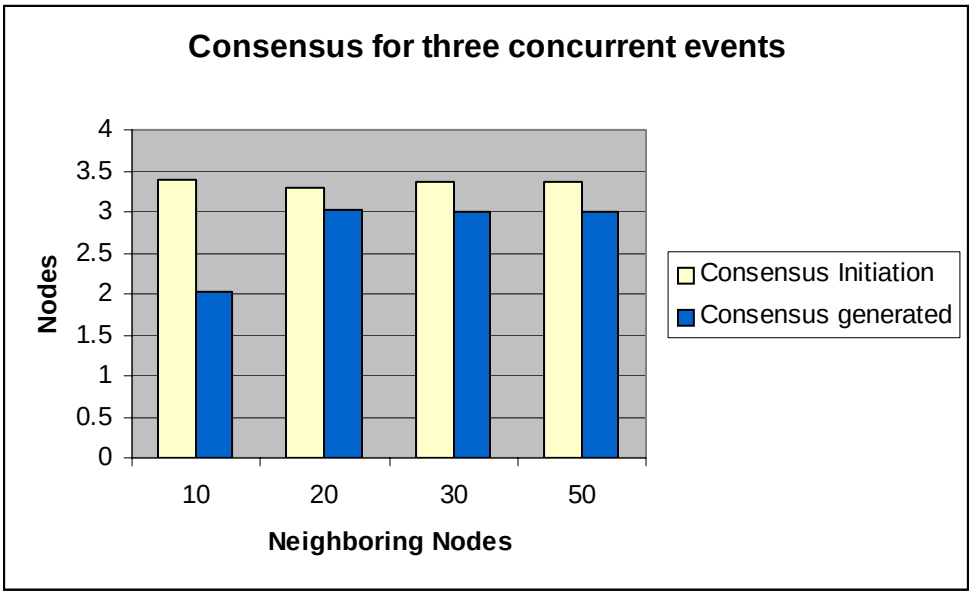

FIGURE X: Variation Of Consensus Initiation For Three Concurrent Events With Node Density

The message complexity of the protocol is linear. A consensus generation process for *k* nodes requires *k+1* messages in the ideal case. The first message is the consensus initiation request, which leads to *k-1* responses, and the last is the delivery of the consensus result to the remaining nodes in the neighborhood. We thus have a linear message complexity in our protocol. The communication overhead in case of higher redundancy is thus encountered only from the reply messages that are received from additional nodes in the neighborhood.

It is observed in Figure X that the consensus was generated for three events in the network. As the sensor network operates as a dense network, the nodes having only ten neighboring nodes were able to generate consensus only for two concurrent events. This was due to the fact that the CVCount of nodes was either four or five. The third event was detected by only one node, which was therefore unable to generate consensus for that event. It was observed however that with an increase in the number of neighboring nodes, multiple independent events were recognized by the sensors, as more than one node was able to witness that event.

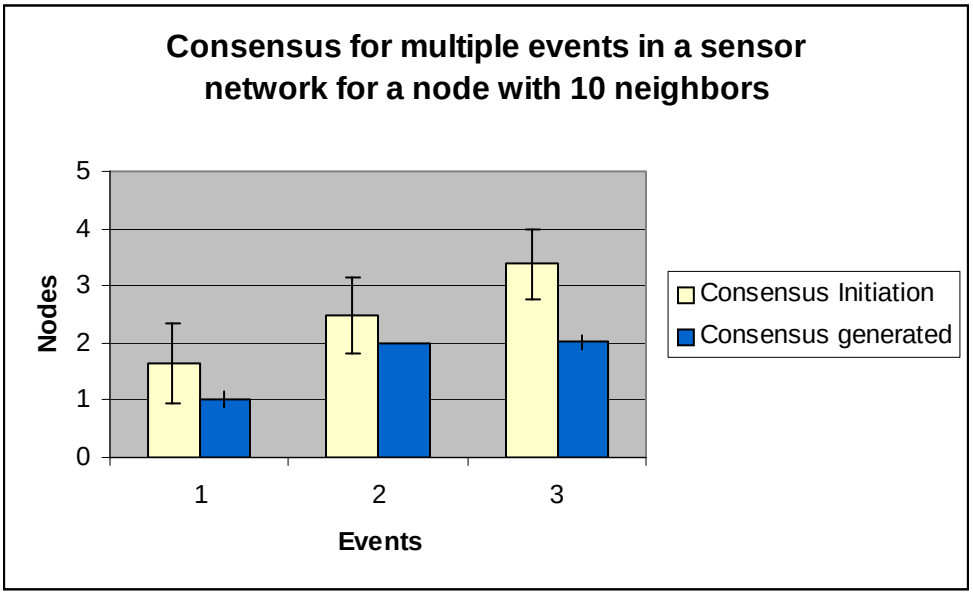

Figure XI: Variation Of Consensus Initiation For Multiple Events With Ten Neighboring Nodes

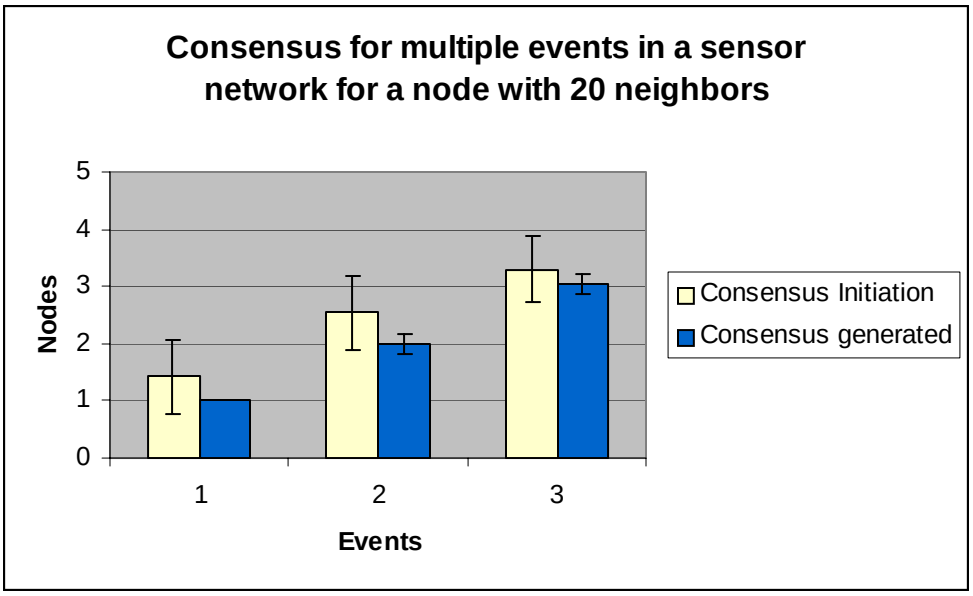

Figure XII: Variation Of Consensus Initiation For Multiple Events With Twenty Neighboring Nodes

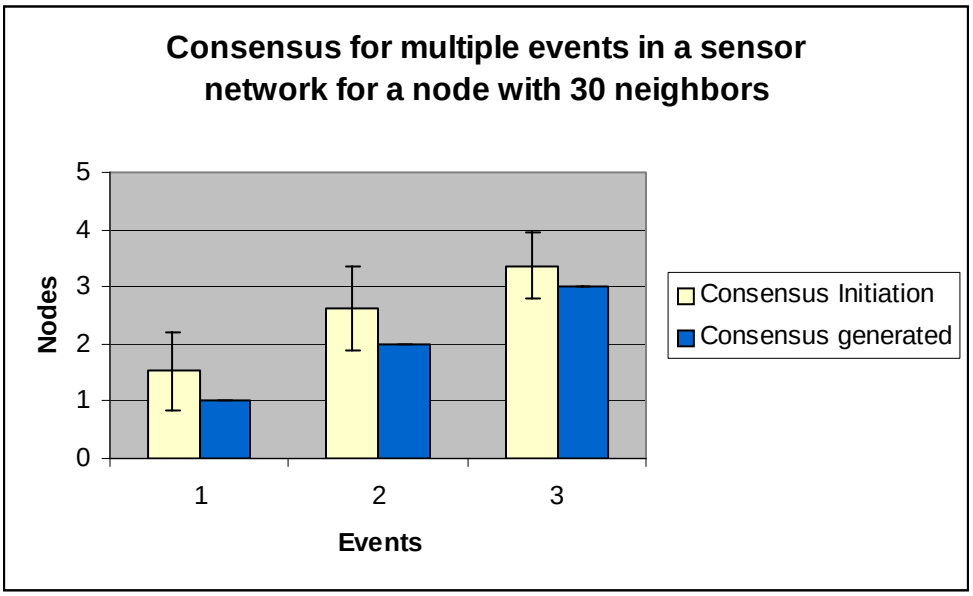

Figure XIII: Variation Of Consensus Initiation For Multiple Events With Thirty Neighboring Nodes

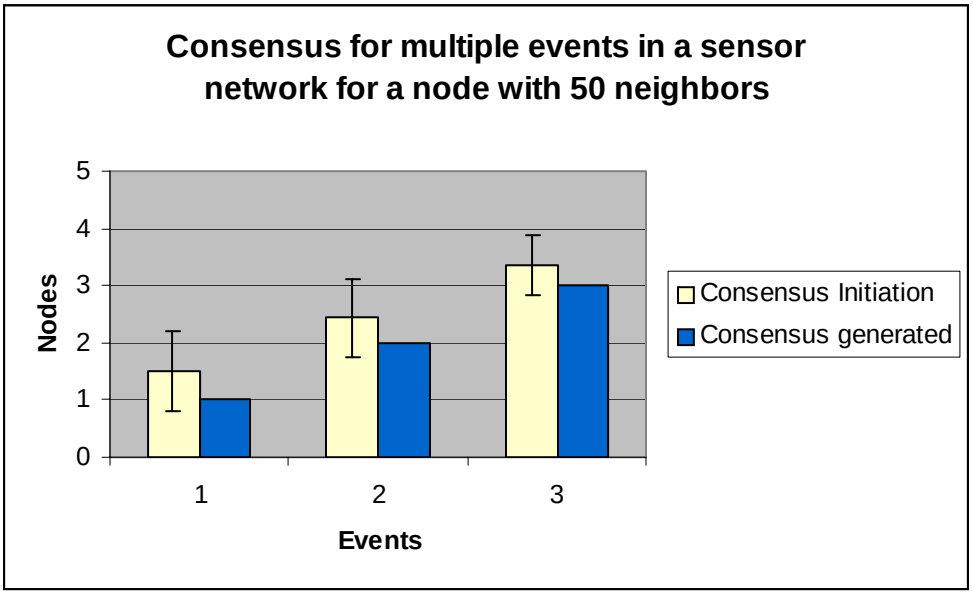

Figure XIV: Variation Of Consensus Initiation For Multiple Events With Fifty Neighboring Nodes

The nodes in the network initiate consensus after a random amount of time. The random function (equations 5,6) employed time as the seed for generating random numbers.

It is observed that the number of nodes initiating consensus decreases with an increase in the value of the product. This leads to further improvement in the energy consumption of the network as the number of messages required for generating consensus decreases linearly as observed in the following table. For instance, observations are made in Table I for a node having ten neighbors. An

increase in the value of the product $k^* N_i$ leads to a decrease in the number of messages required for generating a consensus.

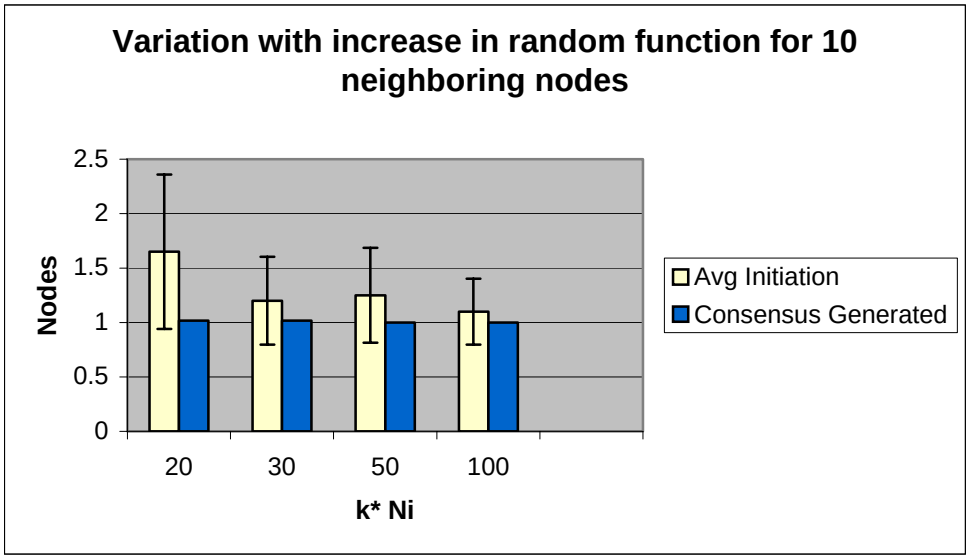

Figure XV: Variation Of Consensus Initiation With An Increase In The Value Of The Random Function For Ten Neighboring Nodes

Table I

Neighboring Nodes	$k^* N_i$	Consensus Initiation	Total Messages	Deviation from Ideal Case	Time
10	20	1.65	18.5	8.5	0.786885
10	30	1.2	14	4	0.852459
10	50	1.25	14.5	4.5	0.913041
10	100	1.1	13	3	1.36667

Variation with the increase in the value of the random function ($k^* N_i$)

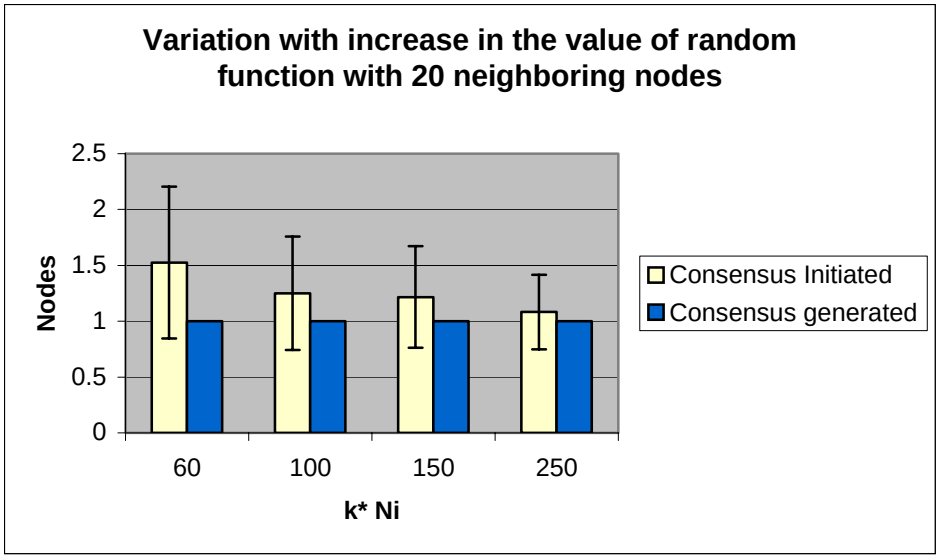

Figure XVI: Variation Of Consensus Initiation With An Increase In The Value Of The Random Function For Twenty Neighboring Nodes

Table II

Neighboring Nodes	$k^* N_i$	Consensus Initiation	Total Messages	Deviation from Ideal Case	Time
20	40	1.41667	16.1667	6.16667	2.05
20	60	1.35	15.5	5.5	2.133
20	80	1.36667	15.6667	5.66667	2.6
20	100	1.2	14	4	2.5

Variation with the increase in the value of the random function ($k^* N_i$) for twenty neighboring nodes

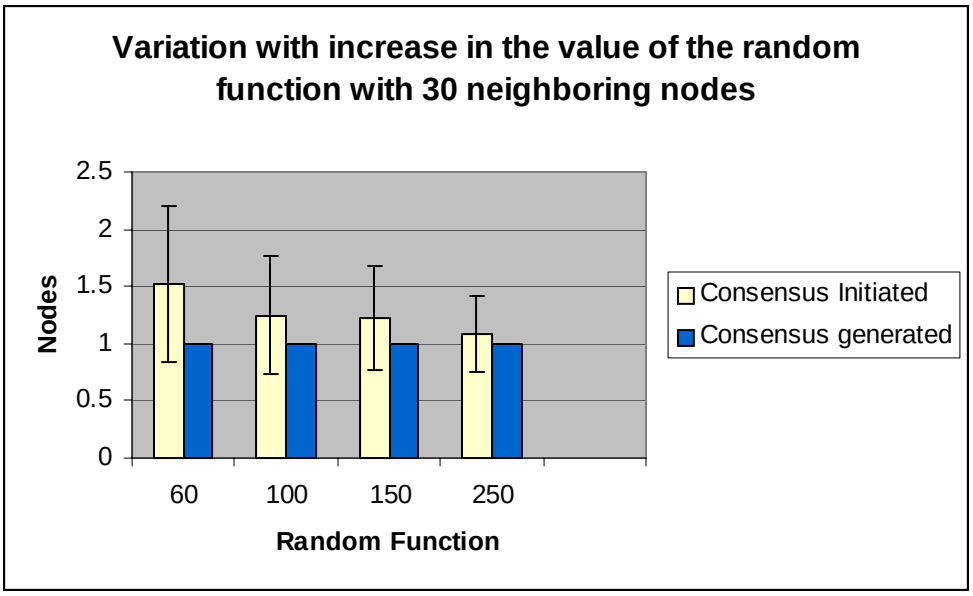

Figure XVII: Variation Of Consensus Initiation With An Increase In The Value Of The Random Function For Thirty Neighboring Nodes

Table III

Neighboring Nodes	$k^* N_i$	Consensus Initiation	Number of Messages	Deviation from Ideal Case	Time
30	60	1.525	17.25	7.25	3.98333
30	100	1.25	14.5	4.5	3.85
30	150	1.216667	14.16667	4.16667	6.76667
30	250	1.083333	12.83333	2.83333	5.95

Variation with the increase in the value of the random function ($k^* N_i$) for twenty neighboring nodes

Latency is another important factor that should be taken into consideration for generation of consensus in a wireless sensor network. There is a trade-off between the time sensitivity of the information and the energy consumption of the network. For instance, consensus should be generated quickly in the case of information like enemy intrusion or forest fires. There can, however, be higher latency in detecting temperature variation within a particular area.

An increase in the value of the product $k^* N_i$ leads to an increase in the time the node would wait before initiating consensus for its information. The following results show the time variation in generating consensus for different values of $k^* N_i$ for a node having the same set of neighbors. A close observation of Figures XVIII, XIX, and XX shows a marginal increase in the latency of the network for an increase in the product of $k^* N_i$. This leads to an improvement in the energy required to generate consensus due to a decrease in the number of nodes initiating consensus in the network (XV, XVI, XVII), which in turn leads to a decrease in the number of messages required for generation of consensus.

We thus observe the trade-off between latency and energy consumption. These parameters can be adjusted based on the requirements of the end user by varying the value of k in the network.

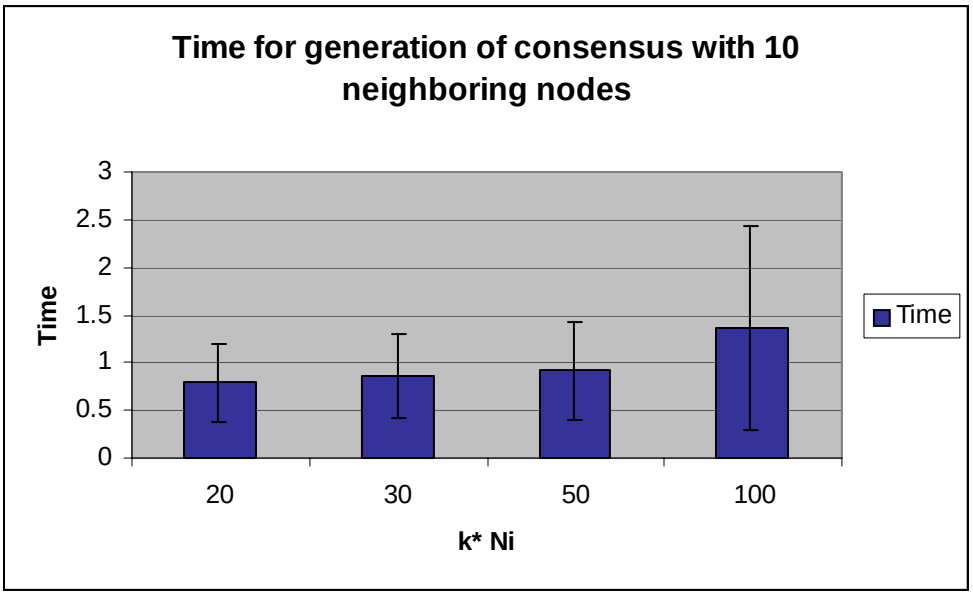

Figure XVIII: Variation Of Latency With An Increase In The Value Of Random Function For Ten Neighboring Nodes.

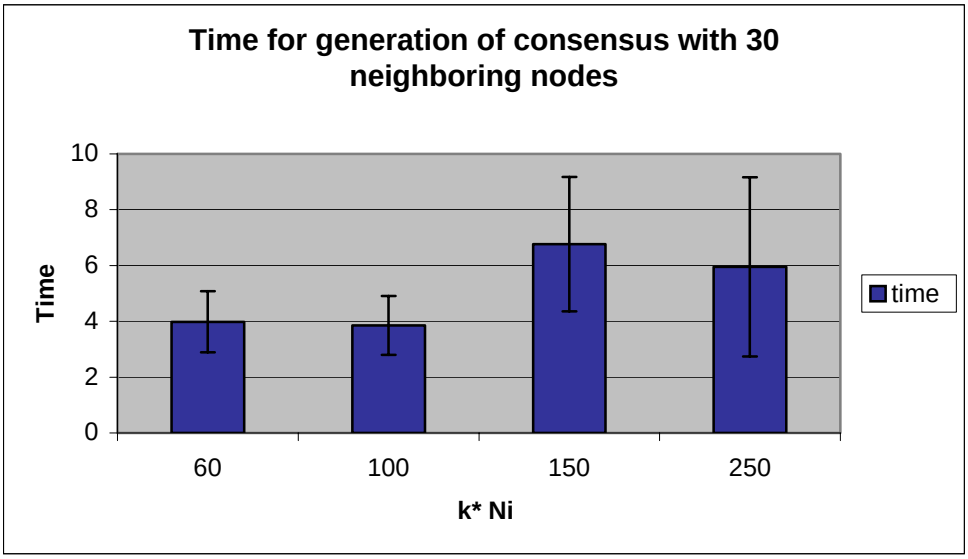

Figure XIX: Variation Of Latency With An Increase In The Value Of Random Function For Thirty Neighboring Nodes.

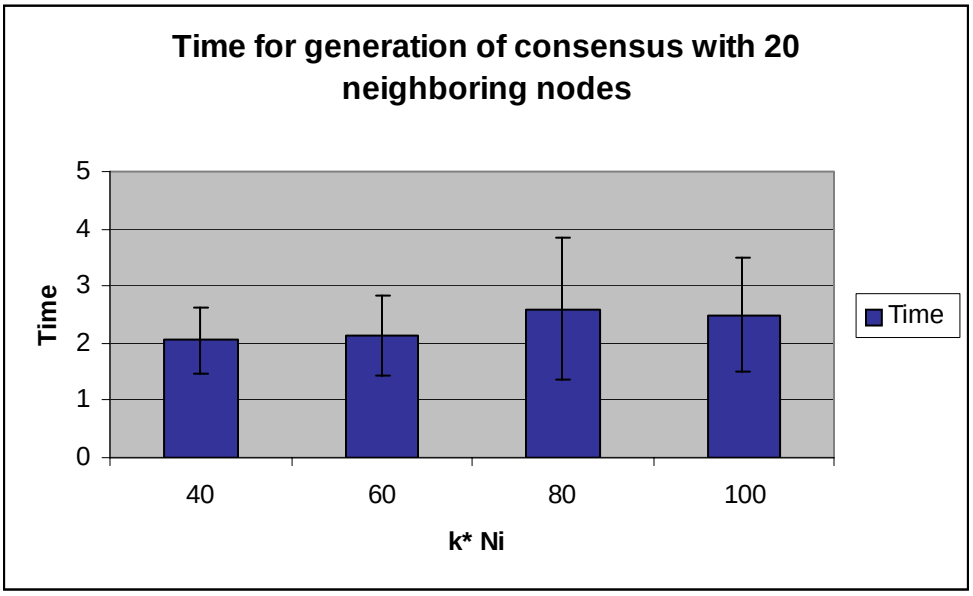

Figure XX: Variation Of Latency With An Increase In The Value Of Random Function For Twenty Neighboring Nodes.

The tabular representation of the above graphs (XVIII, XIX, and XX) is shown in the tables Table I, Table II, and Table III. It can be observed that an increase in the value of $k^* N_i$ leads to a decrease in the number of nodes initiating consensus. For instance, the number of nodes initiating consensus for 10 neighboring nodes and for $k^* N_i$ as 100 is 1.1. As the message complexity for the whole operation is linear, there would be only one extra message that would be required for generation of consensus. The latency for this operation increases by only 0.58 seconds. Similarly, Table II shows that for 20 neighboring nodes the number of nodes initiating consensus are 1.2 for $k^* N_i$ value of 100. It thus requires four additional messages for generating consensus. The efficiency,

simplicity and ease of implementation of our protocol make our protocol really useful in autonomous systems like wireless sensor networks.

When an event is generated in a wireless sensor network, the nodes in the vicinity of the event would start generating consensus for the event after a random amount of time. There can be three different scenarios for initiation of consensus. The first case is represented by Figure VII, where there is only one event in the network neighborhood. The second case as shown in Figure IX involves multiple independent events, which even though they have the same radio range have a different sensitivity range. The third case involves multiple events within the same sensitivity range. This case is shown is shown in Figure XXI. The dark black dots represent the event source. The gray and the blue dots represent the sensor nodes. Proper selection of the random function will enable nodes nearer to the event source (dark blue) to have a higher probability of initiating consensus compared to the nodes (light blue) distant from the source. The nodes will be able to successfully generate consensus for their information as long as the number of nodes having incorrect information is less than the number of nodes having information about the event for which consensus is being generated.

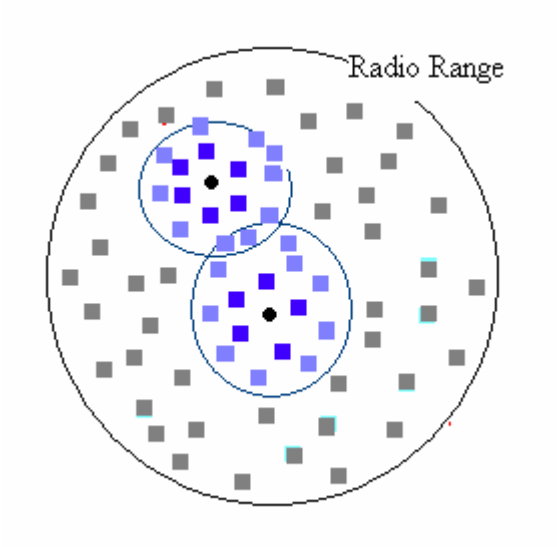

Figure XXI: Concurrent Occurrence Of Events With Overlapping Areas Of Interest

The nodes near the event source have a higher probability of generating consensus compared to the nodes further away from the source. This is represented in the figure by marking the nodes having higher probability of initiating consensus with a darker color as compared to nodes having lesser probability with a lighter color. Our protocol would thus enable the node nearer the event to initiate and generate consensus successfully.

Another important feature of our protocol is fault tolerance. We have performed several tests in order to determine the fault tolerance of our protocol. Our protocol can accommodate up to *(n – 1)/2* faults. Wireless sensor devices are memory-constrained devices and have limited computational and decision

making capabilities. The first sets of results were done for omission and timing failures. The value of *k* has been kept as two for all the remaining tests. .

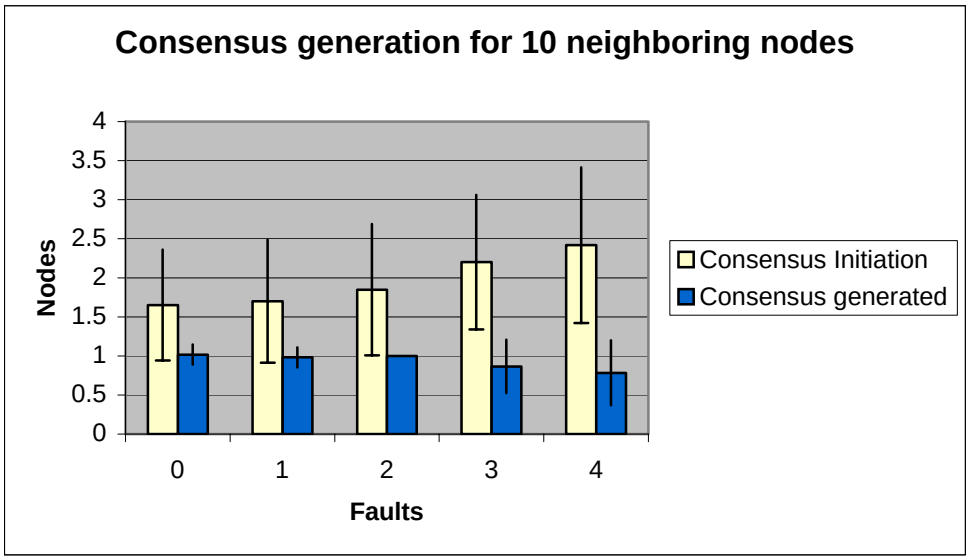

Figure XXII: Variation Of Consensus Initiation For Ten Neighboring Nodes With Different Number Of Faults In The Network

It can be observed in Figures XXII, XXIII, and XXIV that the increase in the number of nodes initiating consensus is reasonably smaller than the increase in the number of faults in the system. For instance the number of nodes initiating consensus for 10 neighbors increases by .77 for an increase in the number faults from 0 to 4 nodes. The number of additional messages for generating consensus would thus only be 7.6667.

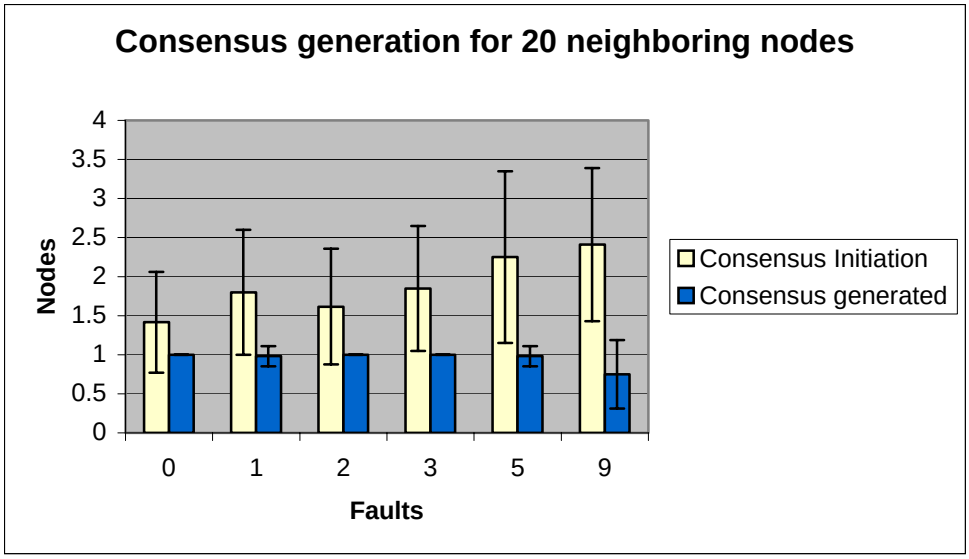

Figure XXIII: Variation Of Consensus Initiation For Twenty Neighboring Nodes With Faults In The Network

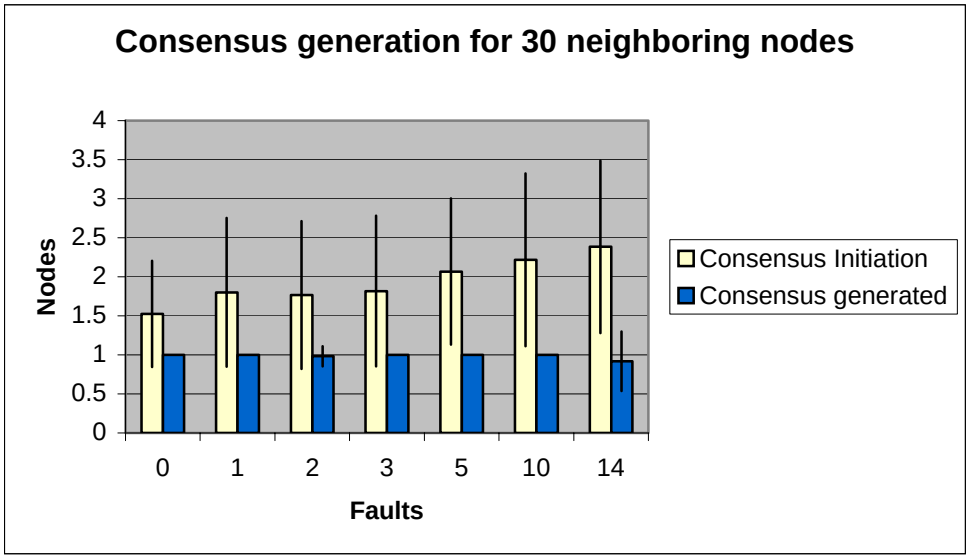

Figure XXIV: Variation Of Consensus Initiation For Thirty Neighboring Nodes With Faults In The Network

Similarly, there is an increase of .86 nodes initiating consensus for sensor nodes consisting of 30 neighbors when the number of faults increases from 0 to 14. This would entail approximately 25 additional messages. In case of a faulty cluster head the number of additional messages required for choosing a cluster head and initiating consensus would be greater than 30 messages. We can thus conclude that our protocol has higher fault tolerance compared to other protocols.

Nodes participating in the consensus generation process can also have different areas of interest. This will lead to different nodes having different values. The following set of results in Figures XXV, XXVI, and XXVII shows the variation with the increase in the number of nodes having different areas of interest for a given set of neighbors.

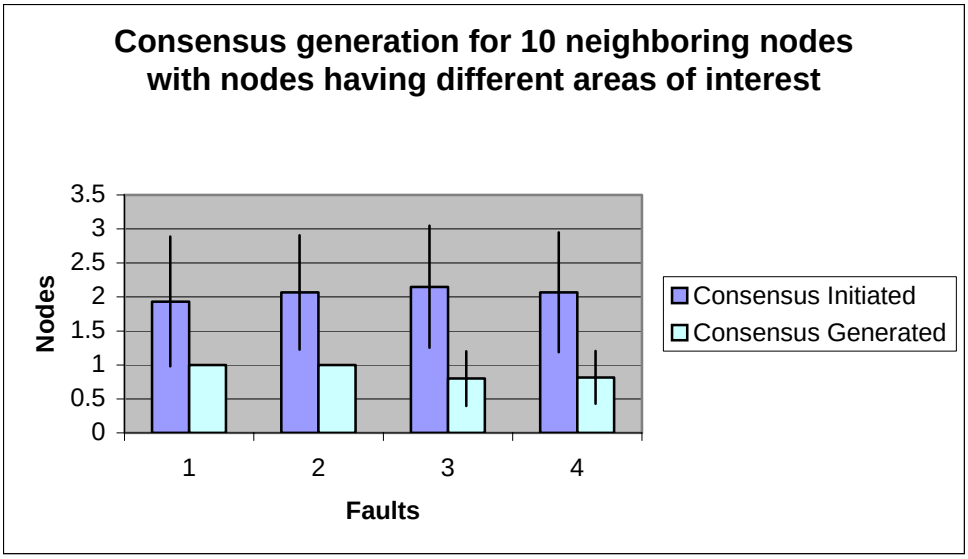

Figure XXV: Variation Of Consensus Initiation For Nodes With Different Areas of Interest For Ten Neighboring Nodes

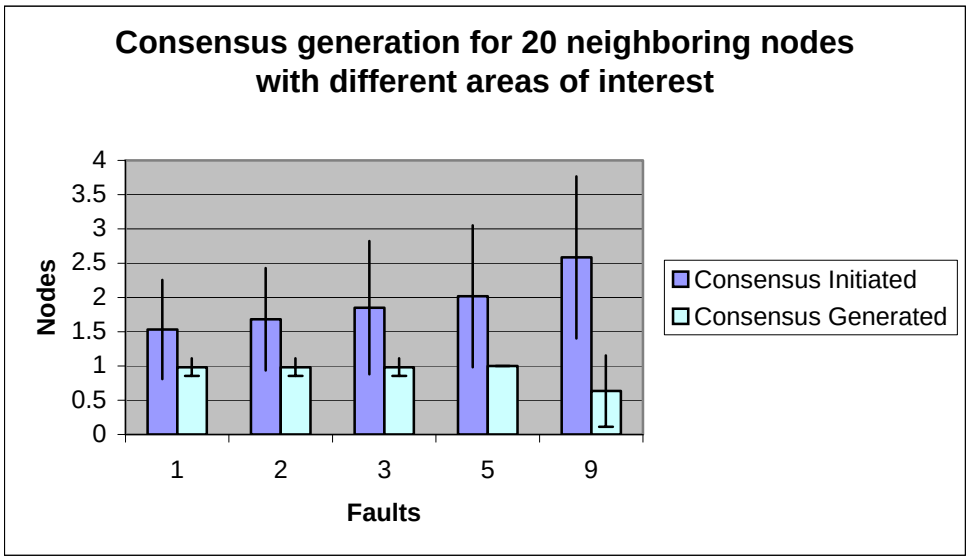

Figure XXVI: Variation Of Consensus Initiation For Nodes With Different Areas of Interest With Twenty Neighboring Nodes

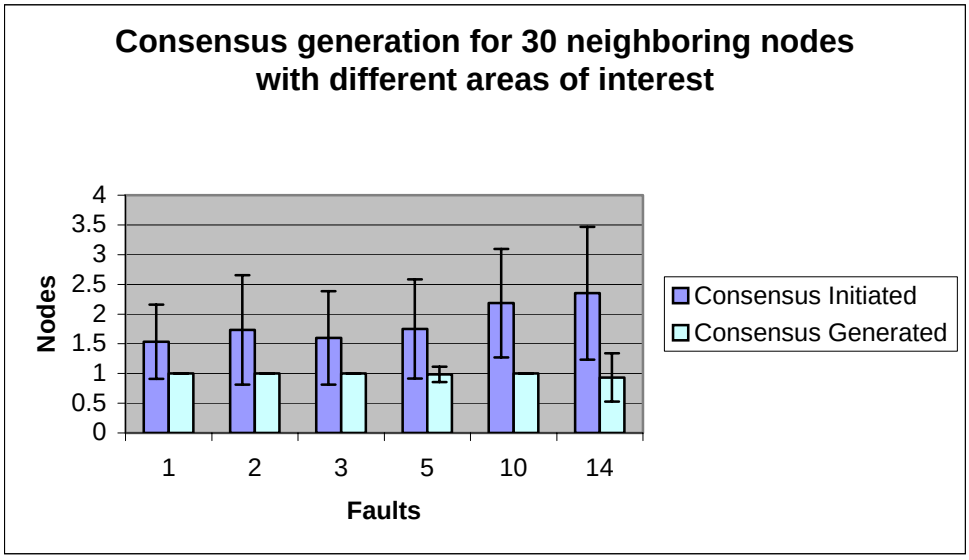

Figure XXVII: Variation Of Consensus Initiation For Nodes With Different Areas of Interest With Thirty Neighboring Nodes

It can be observed that there is only a slight increase in the number of nodes initiating consensus, with the increase in the number of nodes having different areas of interest.

The various experiments enable us to conclude that our protocol works effectively for single as well as multiple events in a sensor network. It is able to generate consensus for concurrent events in a sensor network. It is fully distributed, scalable, and fault tolerant as well as highly energy efficient compared to other protocols.

CHAPTER 9

FUTURE WORK

The variation of the random function for improving the energy consumption of the network is directly related to the latency of the network. Further study needs to be done in order to determine the exact relationship between an increase in the delay in initiating consensus and the consequent reduction in the energy consumption of the network.

We also need to determine the optimal initial probability value p_i to generate consensus by node *i* in the wireless sensor network neighborhood.

We could also look at various scheduling mechanisms that would be required for the nodes to send back message to the base station. It will be difficult to use FDMA, as the cost of hardware would be high for a frequency-hopping device. A variation of TDMA can be used to provide an efficient scheduling mechanism.

We could also extend the protocol to accommodate arbitrary failures. This can be done either by increasing the redundancy in the system or by introducing digital certificates in the sensor network. The tradeoff between an increase in redundancy or an increase in complexity of the network needs to be evaluated.

Future work includes combining the data from multiple groups and then performing data aggregation. This can be optimized using various quorum systems as proposed in [7,8,9,10,11]. The quorum systems discussed by Kumar [11] are hierarchical quorum systems. We can use a variation of this kind of quorum system to extend our work.

BIBLIOGRAPHY

[1] J. Liu, P. Cheung, L. Guibas, and F. Zhao, "A Dual-Space Approach to Tracking and Sensor Management in Wireless Sensor Networks." ACM International Workshop on Wireless Sensor Networks and Applications Workshop, Atlanta, September 2002. Also, Palo Alto Research Center Technical Report P2002-10077, March 2002.

[2] S. Meguerdichian, S. Slijepcevic, V. Karayan, M. Potkonjak, “Localized Algorithms in Wireless Ad-Hoc Networks: Location Discovery and Sensor Exposure”, ACM Symposium on Mobile Ad Hoc Networking & Computing (MOBIHOC), Long Beach, CA, Oct. 4-5, 2001, pp. 106-116.

[3] David K. Gifford. “Weighted Voting for Replicated Data.” In *Proc. Seventh ACM Symposium on Operating Systems Principles*, 1979, pages 150-162, 1979

[4] C. Intanagonwiwat , R. Govindan , D. Estrin,”Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks”. MobiCom 2000, pp. 56-67.

[5] J. M. Kahn , R. H. Katz , K. S. J. Pister, "Next century challenges: mobile networking for “Smart Dust”", MobiCom 1999, pp. 271-278

[6] E.G.Kotsakis and B.H. Pardoe, “Dynamic Quorum Adjustment: A consistency scheme for Replicated Objects”, In *Proceedings of The Third Communication Networks Symposium*, pp. 197-200, Manchester, July 8-9, 1996.

[7] Lorenzo Alvisi, Dahlia Malkhi, Evelyn Pierce, Michael Reiter and Rebecca Wright, “Dynamic Byzantine Quorum Systems”, International Conference on

Dependable Systems and Networks (DSN, FTCS-30 and DCCA-8), New York, 2000

[8] D. Malkhi, M. Reiter, A. Wool and R. Wright, “Probabilistic Byzantine Quorum Systems”, The Information and Computation Journal 170(2), November 2001

[9] Malkhi, Reiter, “Byzantine quorum Systems”, The Journal of Distributed Computing, 11(4):203--213, 1998.

[10] Deepak Ganesan, Bhaskar Krishnamachari, Alec Woo, David Culler, Deborah Estrin, Stephen Wicker, **“**An Empirical Study of Epidemic Algorithms in Large Scale Multihop Wireless Networks”

[11] Akhil Kumar, “A High Availability sqrt{N}Grid Algorithm for Replicated Data.”,

Information Processing Letters 40(6): 311-316 (1991)

[12] Distributed systems: Principles and Paradigms. Andrew S. Tenenbaum, Martin van Steen.

[13] Indranil Gupta, Robbert van Renesse, Kenneth P. Birman., “Scalable Fault-Tolerant Aggregation in Large Process Groups”. 433-442 *Electronic Edition* (IEEE Computer Society DL)

[14] Michael J. Fischer, Nancy Lynch, and Michael s. Paterson. “Impossibility of distributed consensus with one faulty process.” *Journal of the* ACM, 32(2):374-382, April 1985.

[15] Jeremy Elson and Deborah Estrin, “Time Synchronization for Wireless Sensor Networks”, Proceedings of the 2001 International Parallel and Distributed

Processing Symposium (IPDPS), Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing

[16] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "A Survey On Sensor Networks", IEEE Communications, August 2002, pp. 102-114.

[17] G. Hoblos, M. Staroswiecki, A. Aitouche, “Optimal design of fault tolerant sensor networks”, IEEE International Conference on Control Applications, Anchorage, AK, September 2000, pp. 467-472.

[18] 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, July 2001.

[19] Bradley S. Carlson, “Comparison of Modern CCD and CMOS Image Sensor Technologies and Systems for Low Resolution Imaging” IEEE Sensors Conference 2002.

[20] L. Lamport “Time, Clocks and the Ordering of Events in a Distributed System.” *Communications of the ACM*, 21(7):558-565, July 1978.

[21] Peter J Keleher, “Decentralized Replicated-Object Protocols”, 18th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC), April 1999.

[22] Jean-Philippe Martin, Lorenzo Alvisi, Michael Dahlin “Small Byzantine Quorum Systems”, International Conference on Dependable Systems and Networks (DSN'02)

[23] D. Malkhi, M. K. Reiter, and R. N. Wright, “Probabilistic Quorum Systems”. In Symposium on Principles of Distributed Computing (PODC), pages 267--273, 1997.

[24] Indranil Gupta and Kenneth Birman, “Holistic Operations in Large-Scale Sensor Network Systems: a Probabilistic Peer-to-Peer Approach.” International Workshop on Future Directions in Distributed Computing (FuDiCo). June 2002. pp. 1-4.

[25] Stephanie Lindsey, Cauligi S. Raghavendra “PEGASIS: Power-Efficient Gathering in Sensor Information Systems”. IEEE Transactions On Parallel And Distributed Systems, Vol. 13, No. 9, September 2002

[26] Rivka Ladin, Barbara Lishov, L. Shrira and S. Ghemawat. “Providing Availability Using Lazy Replication.” ACM Transactions on Computer Systems 10:4 (Nov. 1992), 360-391.

[27] Jeffrey Hightower, Chris Vakili, Gaetano Borriello, Roy Want “Design and Calibration of the SpotON Ad-Hoc Location Sensing System” (2001)

[28] F. Guerra, S Arevalo, A Alvarez and J. Mirando. “A Distributed Consensus Protocol with a Coordinator”. Decentralized and Distributed Systems 1993: Palma de Mallorca, Spain

[29] Arevalo, S. and Gehani N.H. 1989. “Replica Consensus in Fault Tolerant Concurrent C”. technical Report AT&T Bell Laboratories, Murray Hill, New Jersey 07974.

[30] Lamport, Schostack, and Pease 1982, “The Byzantine Generals Problem”. ACM TOPLAS, V4, N3.

[31] Lamport L. and Fischer, M. 1984, “Byzantine Generals and Transaction Commit Protocols”. Opus 62, SRI International, Menlo Park, CA.

[32] K. S. J. Pister, J.M. Kahn, and B.E. Boser, “Smart dust: Wireless networks of millimeter-scale sensor nodes”, Highlight Article in 1999 Electronics Research Laboratory Research Summary.,1999.

[33] Xiuzhen Cheng Ding-Zhu Du, Lusheng Wang, Baogang Xu, “Relay Sensor Placement in Wireless Sensor Networks”, submitted to *IEEE Transactions on Computers*

[34] L. Schwiebert, S.D.S. Gupta, and J.Weinmann, “Research challenges in wireless networks of biomedical sensors”, *MobiCom 2001*, pp. 151-165

[35] Yi Zou, Krishnendu Chakrabarty, “Sensor Deployment and Target Localization in Distributed Sensor Networks”. *ACM Transaction on Embedded Computing Systems, Special Issue on Networked Embedded Computing: Tools, Architectures and Applications,* 2003

[36]W. Rabiner Heinzelman, A. Chandrakasan, and H. Balakrishnan ``Energy-Efficient Communication Protocol for Wireless Microsensor Networks,'' *Proceedings of the 33rd International Conference on System Sciences (HICSS '00)*, January 2000

[37] Suman Banerjee, Samir Khuller, ”A Clustering Scheme for Hierarchical Control in Multihop Wireless Networks” IEEE Infocom 2001, Anchorage, Alaska, April 2001

[38] G. Colouris, J. Dollimore, T. Kindberg. Distributed Systems, Concepts And Design.

[39] http://www.sensorsmag.com/articles/0303/14/main.shtml Chapter 3., March 2003. Intelligent Systems Power Management “The Impact of Photovoltaic Technology on Sensor Design”

[40] Brett A. Warneke, Michael D. Scott, Brian S. Leibowitz, Lixia Zhou, Colby L. Bellew, J. Alex Chediak†, Joseph M. Kahn, Bernhard E. Boser, Kristofer S.J. Pister, ”An Autonomous 16 mm^3 Solar-Powered Node for Distributed Wireless Sensor Networks”.

[41] Deborah Estrin, Ramesh Govindan, John Heidemann. “Scalable Coordination in Sensor Networks”, Proc. MOBICOM, 1999, Seattle, 263-270.

[42] http://www.xbow.com/Products/Wireless_Sensor_Networks.htm

[43] http://www.sensorsmag.com/articles/0402/40/ Chapter 3, April 2002. Sensor Technology and Design “MICA: The Commercialization of Microsensor Motes”

[44] http://today.cs.berkeley.edu/tos/

[45] Terry *et al* 1995. Terry, D., Theimer, M., Peterson, K., Demers, A, Spreitzer, M. and Hauser, C. (1995). “Managing update conflicts in Bayou, a weakly connected replicated storage system”. *Proceedings of the 15th ACM symposium on Operating Systems Principles,* pp. 172-183.

[46] Peterson et al. 1997. Peterson, K., Spreitzer, M., Terry, D., Theimer, M. and Demers, A. (1997). “Flexible update propagation for weakly consistent replication” *Proceedings of the 16th ACM Symposium on Operating Systems Principles,* pp. 288-301.

ABSTRACT

EFFICIENT AND FAULT TOLERANT DATA AGGREGATION PROTOCOL FOR WIRELESS SENSOR NETWORKS

by

MUKUL KUMAR

December 2003

Advisor: Dr. Loren Schwiebert

Major: Computer Science

Degree: Master of Science

Wireless sensors are versatile, robust, low energy, autonomous devices that are deployed as a dense network in order to gather certain information. The network should have a long operating life and the data gathering protocol should be designed to be energy efficient and fault tolerant. The protocol should also have low latency in order to preserve the relevance of the collected information.

We propose a fully distributed data gathering protocol, which not only reduces the amount of information to be communicated for generation of consensus but also reduces the state information required for maintaining the topology of the network. It has several advantages over contemporary data

gathering algorithms in terms of simplicity of design, efficiency, fault tolerance, and absence of any blind spots in the network.

Each node forms its own group of nodes having common interests based on the sensitivity range of the sensor. A random function enables the nodes near the event source to initiate a consensus, as there is a higher probability of neighboring nodes having witnessed the event thereby improving the probability of the node to achieve consensus for its information. The selection of this random function is also a tradeoff between the latency and the energy required for generation of consensus by the network.

The protocol enables the network to behave as a self-correcting network. Each node stores the probability value of the node to successfully generate consensus for its information. In case the probability value drops below a threshold value, the node tries to form a new group with nodes having similar interests and upon being unsuccessful goes to sleep. This helps in removing the faulty nodes from the system. The protocol accommodates process omission, communication, and timing failures.

The analysis and results prove that the simple and fully distributed nature of our protocol makes it highly efficient and suitable for wireless sensor networks.

AUTOBIOGRAPHICAL STATEMENT

Mukul Kumar

Mukul Kumar is a student of Wayne State University, Detroit, Michigan persevering for a degree in Masters of Science since Fall 2001. He received his Bachelors in Computer Science and Engineering from Panjab University, Chandigarh. His areas of interests include Computer Networks, Software Engineering and Operating System.