Generic Architecture for Power-Aware Routing in Wireless Sensor Networks

A Thesis
Presented to
The Academic Faculty

by

Rishi Ranjan

Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

School of Electrical and Computer Engineering
Georgia Institute of Technology,
Atlanta, Georgia

May 2004

Generic Architecture for Power-Aware Routing in Wireless Sensor Networks

Approved by:

Dr. Ashraf Saad, Thesis Advisor

Dr. A.R.M. Zaghloul

Dr. R. Abler

Date Approved: May 14, 2004

Acknowledgements

I would like to express my deep appreciation to my advisor, Dr. Ashraf Saad, for his support and encouragement. His ideas, comments and suggestions were very invaluable. His guidance was essential in bringing this thesis to completion. I would also like to thank Dr. Saad for his patience in answering all the questions.

I would like to also thank other members of my committee: Dr. A.R.M. Zaghloul and Dr. R. Abler for their time and effort in reviewing my thesis. I found Dr. Abler's comments on presentation style very useful. I am also grateful to Dr. Zaghloul for encouraging me to include more original ideas in the work.

I would also like to thank Dr. R. Badrinath and Dr. R.V. Rajakumar, my academic advisors at Indian Institute of Technology, India, for encouraging me to continue my studies and helping me on my important matters.

Last, but not least, I would like to thank Aruna Srinivasan for her continuous encouragement and my parents, Mithilesh Kumar Thakur and Vibha Thakur, for everything they have done for me.

Table of Contents

List of Figures

Glossary

ARP	Address Resolution Protocol
CEDAR	Core Extraction Distributed Ad Hoc Routing Algorithm
DCF	Distributed Coordination Function
IP	Internet Protocol
LEACH	Low Energy Adaptive Clustering Hierarchy
MAC	Medium Access Control
MDS	Minimum Dominating Set
MMDS	Modified Minimum Dominating Set
PSFQ	Pump Slowly Fetch Quickly protocol
Sink	Receiver in sensor networks
SPIN	Sensor Protocol for Information via Negotiation
TTDD	Two-Tier Data Dissemination
TTL	Time to Live

Summary

This work describes the design and implementation of a generic architecture to provide a collective solution for power-aware routing to a wide range of problems in wireless sensor network environments. Power aware-routing is integral to the proposed solutions for different problems. These solutions try to achieve power-efficient routing specific to the problem domain. This can lead to challenging technical problems and deployment barriers when attempting to integrate the solutions. This work extracts various factors to be considered for a range of problems in wireless sensor networks and provides a generic framework for efficient power-aware routing. The architecture aims to relieve researchers from considering power management in their design. We have identified coupling between sources and sinks as the main factor for different design choices for a range of problems. We developed a core-based hierarchical routing framework for efficient power-aware routing that is used to decouple the sources from sinks. The architecture uses only local interaction for scalability and stability in a dynamic network. The architecture provides core-based query forwarding and data dissemination. It uses data aggregation and query aggregation at core nodes to reduce the amount of data to be transmitted. The architecture can be easily extended to incorporate protocols to provide QoS and security to the applications. We use network simulations to evaluate the performance of cluster formation and energy efficiency of the algorithm. Our results show that energy efficiency of the algorithm is better when the transmission range is kept to a minimum for network connectivity as compared to adjustable transmission range.

Chapter 1

Introduction

1.1 Wireless Sensor Networks

With recent advances in sensor technology, thousands of devices of low computation capalability, communication capability, and low power can be deployed to sense the environment. These sensors are not as reliable and robust as their expensive counterparts, but their size and cost economy enable the deployment of thousands of these sensors in the proximity of the sensed environment, providing a fault-tolerant and high-quality sensing network. Network redundancy provides fault tolerance, while proximity to the target provides high quality since signals decay exponentially with distance [1]. Reliable monitoring is important for many applications, for example, for battlefield, seismic, acoustic, and intrusion detection applications. The data being sensed must be transmitted to the base station, or sink in sensor network terminology, where it can be processed. The source of power to transmit data is typically the battery powering the sensor node, and that has only a limited amount of power. So, the micro-sensor nodes should collaborate to reduce the power consumption to maximize network lifetime. Figure 1.1 shows two sensor nodes developed by Crossbow Technology [21].

Figure 1.1 MICA Motes from Crossbow Technology

Sensor networks typically contain a lot of data to be processed by the sink. Therefore, data aggregation is very important in this environment. It not only avoids overload in the network, but it also reduces redundant information flow, thus reducing overall power consumption. Traditionally, the client/server computing model has been used for multi-sensor data fusion. Mobile agents are seen as an alternative to data fusion of today's sensor networks [2]. In this model, the data stays at a local site and an agent is moved throughout the network to perform data fusion autonomously.

1.2 Problem Statement and Motivation

Several solutions have been proposed for a wide range of challenges faced in wireless sensor networks. Each solution is geared toward solving a particular problem. We have identified the following major problem domains for sensor networks:

- Data Dissemination
- Data Aggregation
- Reliability Transport
- Sink Mobility

- Continuous Query Processing

Solutions for an entire range of problems have been proposed, but each is catered to the individual problem. Due to resource constraints of the sensors, the protocols for handling the problems should be very lightweight. Integrating the proposed solution can become a major technical issue if we try to generate a lightweight protocol. So, we aim to provide a generic architecture that can be used to solve the above-mentioned problem domains.

1.3 Network Topology used in Simulations

The network topology used in simulations is shown in Figure 1.2. It consists of sensor nodes distributed in a 2000m X 2000m area. A total of 400 sensor nodes are distributed uniformly over this area. The nodes are separated by 105m from one another in all directions as shown in figure 1.2.

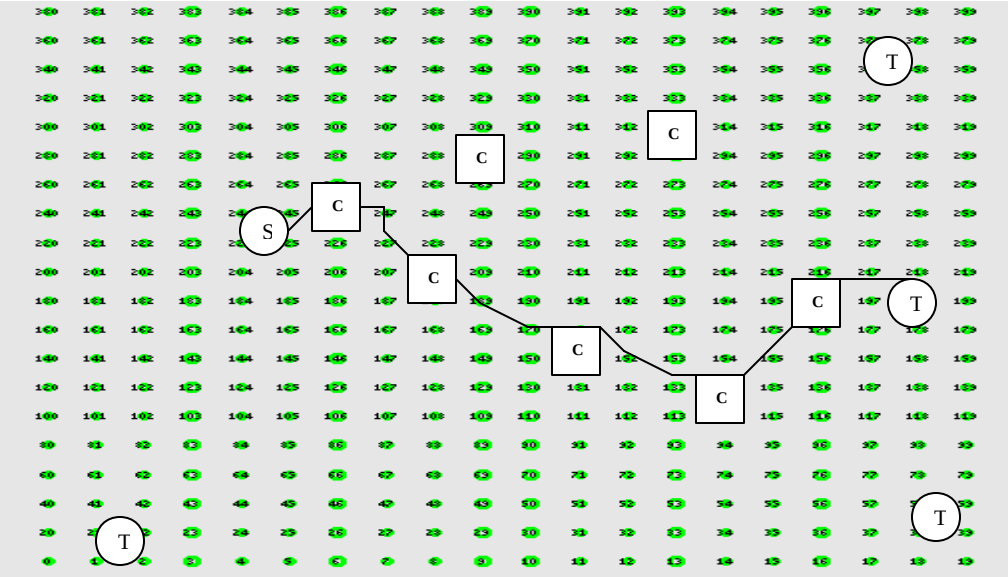

Figure 1.2: Sensor Network Topology.

In figure 1.2, nodes labeled C are elected core nodes. Core nodes are responsible for routing. We call other nodes as common nodes. Common nodes do not perform any routing. They just forward data received from a target or a core node to the next hop nodes on the path. Topology has a sink node S and number of mobile target nodes T.

1.4 Organization

The thesis is structured as follows. Chapter 2 describes related research in the field of power-aware routing in sensor networks. The chapter also describes our approach to solve the problem. Chapter 3 describes the design and implementation of our approach. Performance results and conclusions can be found in Chapter 4. Chapter 5 outlines possible directions of future research. Chapter 6 concludes the work.

Chapter 2

Related Work

2.1 Existing Approaches for Power Awareness in Sensor Networks

Directed diffusion [3] has been proposed for the data dissemination domain. It is motivated by the requirements for robustness, scaling, and energy efficiency in a sensor network. For robustness, the solution proposes the notion of multi-path delivery. The sink keeps on broadcasting the query to its neighbors to check if there is any stimulus. To ensure reliability, the query is periodically broadcasted toward the source. In a special case when the neighbor nodes do not have information available about the source, they again broadcast the query to the neighbors. Directed Diffusion utilizes reinforcing of the data path to pull data on a particular path. The sink reinforces the path to receive data on a path providing the highest data rate.

SPIN [4] is a proposed solution for efficient data dissemination in an energy-constrained environment. The protocol tries to avoid the flow of redundant data in the network to achieve energy efficiency. SPIN [4] eliminates the following problems involved with classical data dissemination networks:

- Implosion: Flooding transmits data to all the neighbors even if they have already received the data. This can lead to the broadcast storm problem [5]. Broadcast has been shown to be very unreliable in ad hoc wireless network environments [5].
- Redundancy: Nodes have overlapping areas. This leads to a large amount of redundant data forwarded on the network.

- Resource Blindness: Traditional approaches do not care about the resource constraints of the system. With sensor networks, energy and computational constraints make an important factor in any design.

Upon occurrence of a stimulus, the nodes send notification to all the neighbors. This is again forwarded to the next neighbor. The protocol performs data aggregation along with data forwarding.

TTDD [6] is another novel approach dealing with data dissemination. The source forms a hierarchical virtual infrastructure to disseminate the data to the sinks. The protocol is designed to solve the problems arising as a result of sink mobility. It has been shown that the protocol is very efficient and robust in dealing with this problem. The protocol is also energy-efficient in this situation.

PSFQ [7] is a proposed solution for a reliable transport protocol. There is an emerging need for reliable transport protocols in wireless sensor networks. Because of the application-specific needs of the sensor networks, it is difficult to design a single generic transport protocol. PSFQ [7] tries to provide a transport protocol that is customized to the needs of different applications. Examples of applications that need reliable data delivery is reprogramming and re-tasking the sensors. This may be required to change the mission of the sensor network. However this requires reliable data delivery support from the underlying transport layer. Such devices that can be programmed over the network already exist, for example, Berkeley Motes [8]. They can receive code segments from the network and assemble them to program a sensor for a new task. Normally, data sent from source to sink is loss-tolerant because of redundancy. But control and management information sent from the sink to the source are loss-sensitive. A

slight loss of data can render the entire code image useless. The main challenge is in programming a group of sensors, loading binaries, and switching to a new application. PSFQ [7] tries to ensure reliable data delivery to all the receivers with minimum support from the underlying infrastructure. It tries to minimize the number of data transmissions per lost packet and handling recovery operations with minimum signaling. PSFQ [7] operates very efficiently even in an environment where link quality is very bad. The approach tries to solve the problem of reliable data transport in order to get maximum energy efficiency.

Continuous query processing [9, 10] tries to solve the problem of a long-lived query in the network. It tries to apply a traditional database query-processing approach to sensor query processing. This research also tries to keep energy consumption as low as possible.

All proposed solutions try to achieve high energy efficiency and data throughput or low latency for a particular problem domain.

2.2 Proposed Architecture

After careful analysis of the different problem domains, we identified two main causes of the existing problems in the sensor network's environment:

1. Lack of routing infrastructure
2. Coupling of source and sink

First, we give a brief overview of our proposed architecture and then discuss the deficiencies in other proposed solutions and show how our scheme is used to overcome these deficiencies.

After a thorough analysis of the problem domains, we propose a core-based architecture for sensor network environments that provides an efficient solution for a wide range of problems in this domain. Using this infrastructure, we decouple the source and sink and provide a virtual routing infrastructure. The key features of the protocol are:

- formation of a core infrastructure of a minimum dominating set of nodes,
- rotation of the core nodes,
- local data fusion at the core nodes, and
- specific components for each problem domain utilizing the core for efficient operation.

We use the following network model to evaluate our approach:

- The network comprises a large number of redundant identical sensor nodes that communicate over radio.
- Range of radio can be adopted for topology control of the core.
- Once a stimulus occurs in the network, sensors surrounding it will collectively generate data and report it to the respective core nodes.
- We assume a first-order radio model, as proposed in LEACH [11]. It is a simple model that assumes that radio dissipates E_{elec} energy per bit in the transmitter or receiver circuitry and E_{amp} energy per bit per unit surface in the transmission amplifier in order to achieve an acceptable signal-to-noise ratio.
- All nodes in the cluster communicate with each other on the same, shared wireless channel. Assuming Direct Sequence Spread Spectrum (DSSS), all nodes have the same pseudo-random sequence.
- We are using 802.11b as the MAC layer protocol [12].

The driving force behind providing a backbone infrastructure for sensor networks is the fact that communication is more expensive than computation from a power consumption standpoint. Therefore, we want the data transmission activities to be performed by a minimal set of nodes. The other goal is to avoid the broadcast storm problem. At the same time, we want the number of nodes in the core to be optimal (i.e., kept to a minimum) in order to maximize network lifetime, similar to the analysis performed by LEACH [11].

We chose the modified version of the minimum dominating set (MDS), concept from graph theory, to form the core of the network. The reason behind using MDS is that we want to ensure that the core provides infrastructure services to all nodes in the network. The dominating set for a given graph is defined as a set of the graph whose union with its neighbors constitutes all elements in the graph. Thus, a network with a dominating set of nodes as its core can provide service to all the nodes in one hop. We chose an alternative definition for the minimum dominating set that suits our architecture. We call it the modified minimum dominating set (MMDS), and defined as follows.

Definition:

Modified Dominating Set (MDS): Given a graph G = (V, E), where V are the vertices and E the edges comprising the graph, we define MDS $V \subset G$ such that every node in G is either in V or is a k-hop neighbor of a node in V. MDS with minimum cardinality is the modified minimum dominating set (MMDS). Given MMDS V of a graph G, we define the core of the graph C = (V, E), where E = {[u, v] | $u \in V$, $v \in V$, and $u \in S$. S(u) = Set of nodes whose distance from u is not greater than k, excluding the node u.

Once core node is elected, maintenance of the corresponding cluster is restricted to local communications only. Rotation of core nodes within the cluster is used to elect new core nodes. This will ensure that network-wide interaction for cluster formation is needed only at the time of network deployment. After that, local nodes collaborate to maintain the cluster. The core node in a cluster adapts its radio range as per the topology requirement of the core. An energy resource manager will decide whether a node can become a core node. The decision is made through local interactions within the cluster. Once the energy level of a node goes below a predefined threshold level, the energy resource manager will remove the node from participating in the core node election. This implies that if the energy resource manager decides that a certain node cannot perform the tasks of a core node, the node will not be allowed to become a core node.

Core node rotation plays a very important role in the architecture. The core node communicates at a high power level with other core nodes, which may drain out its power quickly. Apart from this, our approach adds more intelligence and responsibility to the core nodes. This will also consume more power from a node. Core nodes are then rotated periodically so that energy is dissipated evenly from all nodes within a cluster, hence maximizing network lifetime.

A simple model is being used for data fusion that focuses on redundant data elimination. When a core node receives data from different sensors in the network, it compares the data to check for any redundant information. To uniquely identify the data, we use the sequence number and the network id contained in each packet header. The source nodes include their respective network id and a sequence number in the packet

header. We discuss next the deficiencies of other proposed solutions that we used to evaluate our architecture.

In directed diffusion, if sensor nodes do not have any idea about the location of the source node, they rebroadcast the query again. This can lead to the broadcast storm problem. Our architecture never attempts broadcast by virtue of setting up a core network. The core nodes have all the information about the coverage area and the core topology. Thus, any query can be delivered to the source node through unicast.

In directed diffusion the sink keeps on broadcasting the query to the source nodes to check for any stimulus. Our architecture does not depend on passive query from the sink. For data dissemination, as soon as a node senses any stimulus, instead of waiting passively for a query from the sink, it will proceed to update the respective core node about the stimulus. The information will then be forwarded to the entire core. Thus, we save on the formation and maintenance of the dissemination infrastructure as the core is formed once at the deployment of the network. We also achieve considerable energy savings by avoiding continuous query broadcasts from the sink node as broadcast has already been proven to be very unreliable in ad hoc wireless network environments [5]. Passive query also forces all nodes in the neighborhood of the sink to be involved in query forwarding. But in our case, only core nodes are involved in query processing. Since all the stimulus information is already located at the core, the sink just needs to query the respective core node. The other deficiency in direct diffusion is the enforcing concept. The sink enforces a high data rate path to retrieve data. This is not suitable for sensor network type-environments. The nodes on the enforced path will be drained at a very fast rate, which may result in network partitioning. This is also a direct effect of

coupling between the source and sink. Once we decouple the source and sink by a core network, it automatically takes care of this inefficiency. Since rotation of core nodes dissipates power from all the nodes uniformly, our approach can lead to increased network lifetime. A similar approach is taken by TTDD [6], but its data-forwarding infrastructure is very simple. There is still room to improve on the backbone infrastructure topology. The core tries to improve on this fact. This can be very important for sensor networks processing real-time data.

A node in SPIN [4] transmits data to all its neighbors for data dissemination. This can be a very resource-consuming and inefficient strategy for data dissemination since in a typical sensor network only the sink is interested in receiving the data. Other nodes just provide support to transport the data. So, a minimum number of nodes should be used to disseminate data to ensure energy efficiency. Our approach leverages the already constructed infrastructure of the core. Therefore, data will be transferred from the local cluster node to the core, and the core will forward the data to the sink's core node. Therefore we try to involve a minimum number of nodes in energy-consuming activities. The core node at the source end acts as a very effective data-aggregating agent. We do not want to use mobile agent-based data aggregation because of security problems involved with it. Since all data from a cluster passes through a core node, the core node can drop the redundant data packets, thereby increasing energy efficiency. We want to point out that this aggregation is apart from what is performed at the local cluster nodes. Thus, this comes in addition to what SPIN [4] can achieve. This can be achieved only with a backbone infrastructure *overlayed* over the sensor network.

PSFQ [7] approach argues that overhead involved in transmitting data through the entire network is justified since the intended receiver is the entire network. But in most real-world problems, only a group of sensors is to be reprogrammed. The scenario where the entire network is involved for transmission can be very resource consuming. Since our architecture uses a core-based scheme, we can localize reliable data delivery to the nodes in a given cluster only. Thus, data will be transferred reliably to the respective core node through the core infrastructure and the core node can then forward the data within the cluster for reliable delivery.

In LEACH [11], a similar hierarchical routing protocol has been analyzed, but the approach only aims to maximize the lifetime of the network. Cluster-heads were responsible only for transmitting the data to the base station. Our approach makes the core nodes more intelligent in the sense that the nodes collaborate with each other to accomplish a wide range of tasks. The backbone infrastructure created by the core nodes can be used by the already proposed solutions, as they will be applied by the core nodes in the local domain only. We want to emphasize that though our architecture is self-sufficient to handle a range of problems, the already proposed solutions can definitely gain from services provided by our architecture. Also, LEACH hierarchical cluster maintenance involves network-wide interaction of the nodes. Our approach has an advantage over it being a totally localized one.

Chapter 3

Design and Implementation

3.1 Core Formation

Since computing a dominating set of a graph is an NP-hard problem, we have used the approximation of the algorithm from CEDAR [14] to compute the minimum dominating set of a given sensor network. The algorithm is based strictly on local interactions. We have identified the number of hop "k" in a local cluster as a design metric. We have implemented the algorithm in ns2 [15]. Once a core node has been elected through local interaction, it has knowledge about all neighbors in its cluster. This information is used by a core node to transmit data to neighboring core nodes and local cluster nodes.

The following messages are exchanged between local nodes to elect core nodes. After message exchanges are over, core infrastructure is established and each common sensor nodes associates itself to a core node.

Initial Hello Messages for Neighbor Discovery: When simulation begins all nodes start broadcasting HELLO messages at random intervals between 0 and OUR_ALGO_MAX_HELLO_INTERVAL (0.1 second). These hello messages are broadcasted for hello_period_(5 seconds). This gives every node an opportunity to broadcast the hello message 50 times. This ensures that if the message gets destroyed due to collision, it will be reliably transmitted to all the neighbors through multiple

broadcasts. Here we assume that the sensor nodes are stationary, so the HELLO message exchange is only done initially during the network self-configuration.

GEN_POW_CORE_NODE message exchange for core discovery: After neighbor discovery, each node starts broadcasting GEN_POW_CORE_NODE messages at random intervals between 0 and OUR_ALGO_MAX_HELLO_INTERVAL (0.1 second). Following are the messages broadcasted for the next hello_period_(5 seconds) :

1) The message contains (u, N (u), N*(u), dom(u)) tuple for each node where u is the node's address, N(u) is the number of its neighbors, N*(u) is the number of other nodes for which it is acting as a core, dom(u) is its dominator.
2) If a node receives a GEN_POW_CORE_NODE and it has N*(u) greater than zero, the node selects the sender as the core node. This ensures that if a core node exists in a neighborhood, nodes in that neighborhood will select it as their core node.
3) Once a node has received GEN_POW_CORE_NODE messages from all its neighbors and it still did not choose a core node, it selects the node with the maximum number of neighbors as the core node. This ensures that the node having the maximum degree of coverage is selected as the core node.
4) Once a node selects another node as its core node, it sends it a GEN_POW_NEIGHBOR_CORE_NODE message to confirm the selection.

GEN_POW_NEIGHBOR_CORE_NODE message transmission to discover the neighborhood topology by a core node: GEN_POW_NEIGHBOR_CORE_NODE has (w, dom(w)) as its contents, where w is the neighbor of the node transmitting the message and dom(w) is the dominator of w. Each node sends this information for all its neighbors to the selected core node. Once a core node receives this message, it selects itself as its own core node. This message exchange provides the core node with the topological information about neighboring core nodes.

Here we want to stress the point that we have deviated from the CEDAR [14] algorithm for core formation since it exhibits some deficiency in our scenario. Our main contribution is that we do not allow a node to elect itself as a core node once it finds itself as a node having the maximum degree of neighbors among its neighboring nodes. Intuitively it seems that this step may lead to more time for the core formation algorithm to converge since core node election has been delayed. But we argue that this modification will remove any chances of nodes that do not have any sensor nodes attached to them from being elected as core nodes. Since by the time a node elects itself as a core node, it is possible that all the neighbors have already been selected as core nodes. This can be attributed to the distributed nature of the algorithm. Even if a node announces itself as a core node, none of the neighboring nodes will associate themselves with it. We have therefore taken the approach of association by the sensor node. Thus only a common sensor node can decide to elect a node as a core node. With this approach, we guarantee that every core node represents at least one common sensor node.

As shown in Figure 3.1, the central node elects itself as core node and announces it to the neighboring nodes. But the neighboring nodes are already elected as core nodes, thus the central node is a core node with none of the common nodes associated with it.

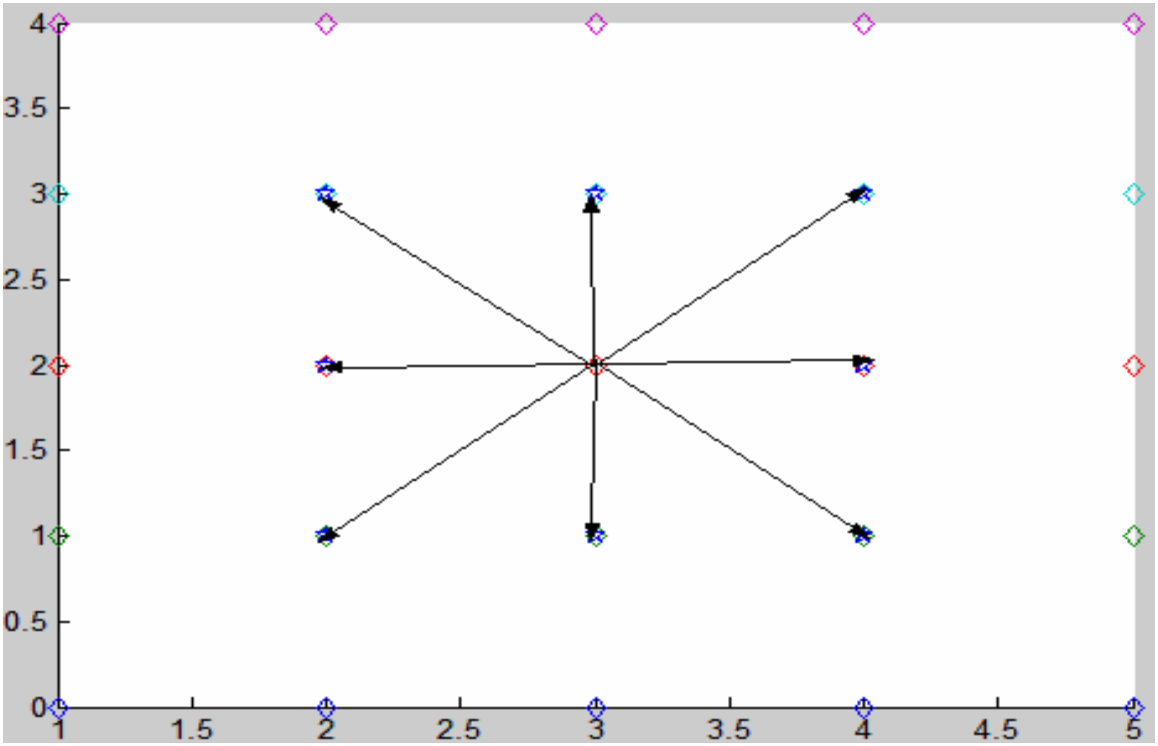

Figure 3.1: Node electing itself as a core node.

3.2 Core Update

We worked on a sensor network model suitable for proactive sensing needed by many applications such as real time monitoring of the environment. This model will be

inefficient for applications where continuous environment monitoring is not essential, like event-based detection of a target.

Once a core node receives a query for an object sensed in its coverage area or in that of its neighboring nodes, it forwards the query to the nodes that sensed the object, and proactively forwards the information to the core node (if other than itself). The sensor node sensing a target proactively keeps on reporting to its core node about the target at a low frequency. This message is forwarded to all the core nodes for maintaining a *soft state* [19] for the existence of information about the target.

During the core formation phase, each core node stores and updates the information about its neighboring core nodes. This information is stored in a cache. When a sensor node detects a target, it forwards the information about the target to its core node proactively. All the sensing nodes report this information at a relatively small rate initially. The basic idea is to propagate this information to all core nodes once the target is detected. This ensures that the core infrastructure has complete information about existing targets in the sensor field. The nodes can update the core infrastructure at the same rate as they sense the target, but it will be a lot of overhead in the case where none of the sinks are interested in the target being sensed. And we want to send data only on the path from which a sink is requesting the data. Data being sent on the other paths will just add to the overhead. The target is characterized by a unique "id." The algorithms to generate this unique "id" are out of the scope of this work. But we propose to use a target's fingerprint being generated by some suitable signal processing technique as the "id" for the target. In our work, targets broadcast their network address to the environment. We use this as the "id" of each node in the simulation, as each node has a

unique network address. Though not a reasonable assumption in an unfriendly environment, the architecture can be extended with an appropriate algorithm that can generate fingerprint for the target.

The update message forwarded to the neighboring core nodes: The update message header contains a unique "id" for the target. The update message also contains a unique sequence number used to characterize the update message for a particular target. The update message header contains a 32-bit entry for this sequence number. The sequence number wraps back to 0 after it reaches 2^{32} -1. It is important that each target maintains its own sequence number. Thus the tuple of <target_id, sequence_number> can characterize a unique update message generated from the same target. The update message also contains source route information to the target it represents. It is important that the length of the sequence number field be large enough to avoid wrap up of the sequence number before the message reaches remote core nodes in the core infrastructure. If the sequence number wraps up very fast, there will be incoherent information in the core infrastructure since remote core nodes may store a stale sequence number similar to that stored at nearby core nodes. We therefore chose a 32-bit sequence number for this purpose. The nodes forwarding the message add their address and their remaining energy level to the source route entry of the packet. Core nodes update the entry for the target id and the sequence number into their cache before forwarding the packet to neighboring core nodes. They update the path for this entry as described below. Upon receiving the update message for a new target node, the core nodes add the entry into their respective caches. This packet is then forwarded to the neighboring core nodes

except the one that forwarded the update message. The sequence number is used to differentiate old update messages from new ones. This is in turn used by core nodes to maintain soft states for the targets. If a new update message arrives at a core node, it checks if it already has an entry for the corresponding target node. Depending on the existence of this entry it takes one of the following actions:

1) If no entry is found, it adds the entry into its cache and updates the sequence number for the target. It then adds the weighted mean of the source path length contained in the message and the inverse of the minimum of remaining energy of nodes on source path [18]. Basically, we are using the shortest path route to disseminate both data and query. However, in our case the shortest path is a function of the number of hops, which is the length of the source route path, and the minimum energy of nodes on a path. The total path length is directly proportional to the number of hops, thus shorter hops will be preferred over longer ones. At the same time, the inverse relation of path length to the minimum remaining energy on a path will discourage storing a path with bottleneck nodes even if such a path is shortest in terms of the number of hops. After adding this path to the cache, the core node will forward the update message to all the neighboring core nodes except the one that originally forwarded the message.
2) If the node already has an entry for the given target node, it checks the sequence number for the new packet and takes action as follows:
    a. If the new sequence number is greater than the sequence number stored in the cache, the cache is updated with the new packet information. The sequence number is updated, and the existing path is replaced by the new

path. The message is forwarded to all neighbors of the core node except the one that originally forwarded the message.

b. If the new packet sequence number is smaller than the one stored in the cache, the update message is simply dropped without any processing.

c. If the packet sequence number is equal to the one stored in the cache, the core node compares the path length of the route path in the new message to the one stored in the cache. If the new path is greater than the cached one, the message is dropped without any processing, otherwise, the cache is updated with the new path and the message is forwarded to all the core nodes except the one that originally forwarded the message.

Thus our core updating algorithm supports discovering efficient paths to the target but it suppresses the progress of a less efficient path over the existing one. As a result, inefficient paths are suppressed, saving energy from being wasted and, at the same time, better paths are supported over the core infrastructure.

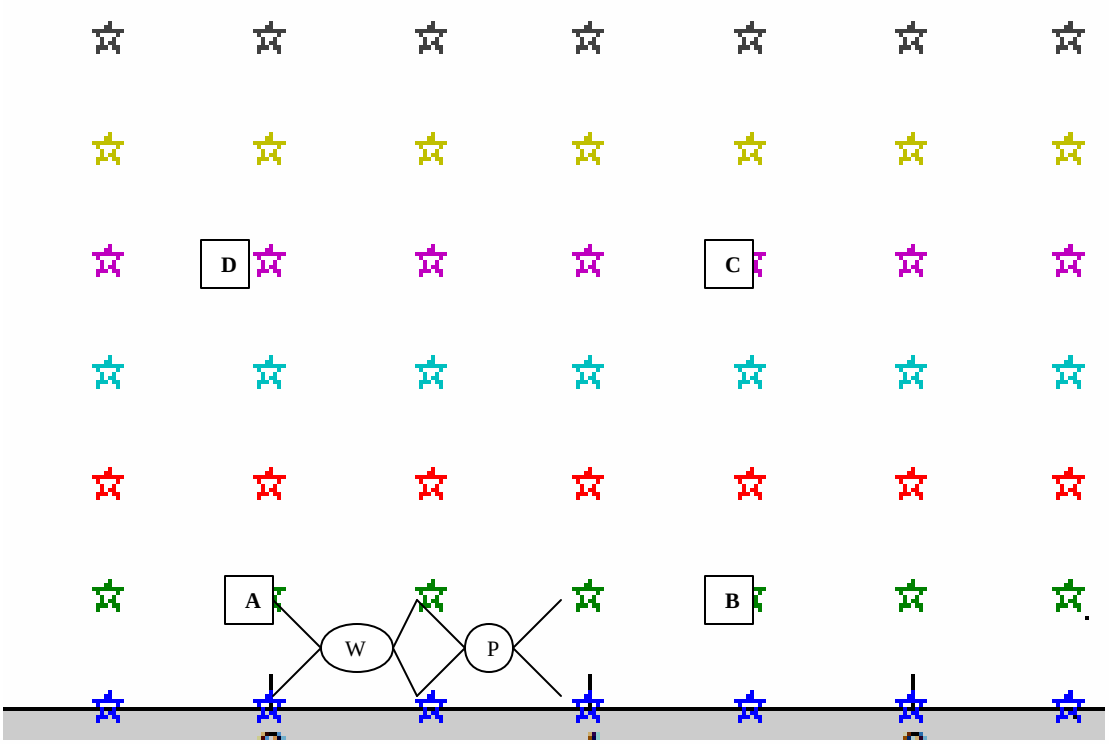

Figure 3.2: Local core update message.

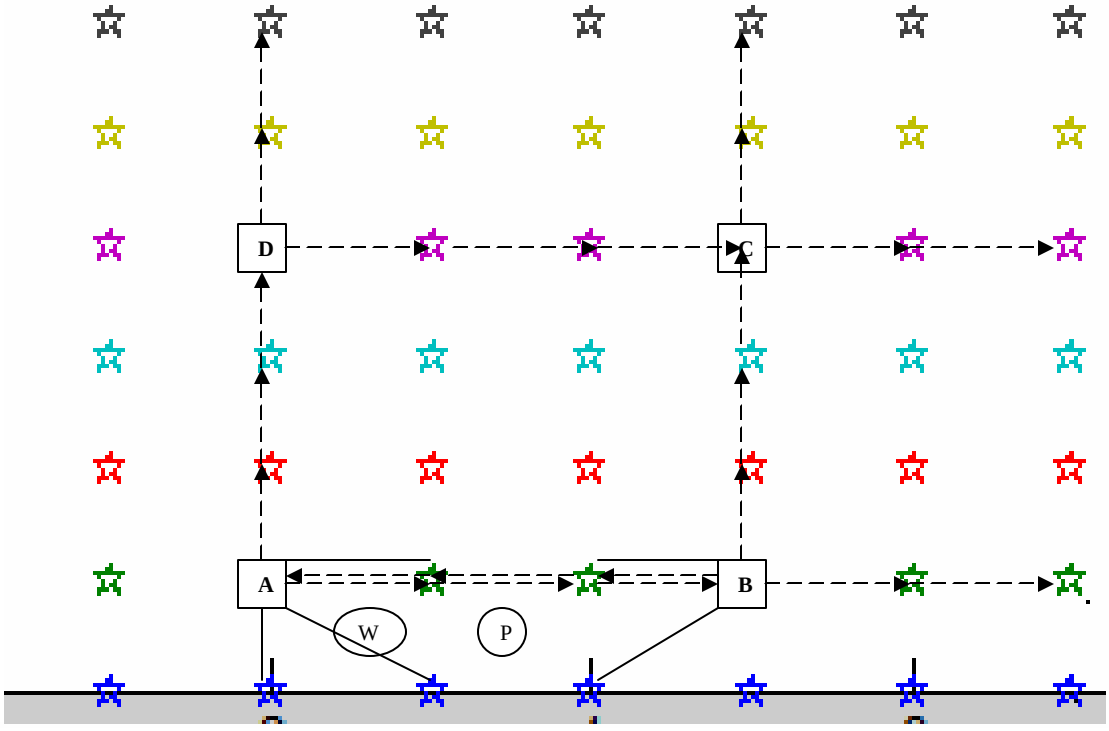

Figure 3.3: Global Core Update.

Figure 3.2 shows the local handling of the core update message. P and W are the target elements to be sensed. Handling of local core update messages has been chosen to be time-driven. Therefore, nodes sensing the targets just keep on updating the most recent sequence number for each target id. This information is not forwarded to the core nodes until a timer associated to the nodes for local updates expires. Thus, even though core nodes A and B keep on receiving the update message, they wait for the timer expiration before forwarding the packet. The local update timer at core nodes are set to higher values as compared to common nodes. Common sensing nodes send the global core update message with short random delay to their respective core nodes. Transmitting at short random delay ensures that transmission is not synchronized leading to collisions. The core nodes higher timer values ensure that the core nodes have all the updates from neighboring common sensor nodes before transmitting a global update message to the core infrastructure. Therefore, in Figure 3.3, common core nodes transmit the most recent target ids to the respective core nodes A and B with short randomized delay after the expiration of the timer. Core nodes A and B have already received local update messages from common nodes by the time their respective timers expire. The core nodes transmit a global update message only if there is a change in the status of stored target ids. Accordingly, core nodes C and D do not forward any global update message to the core infrastructure on their own if they do not have status change after the expiration of the timer. Thus the time-driven model used at core nodes ensures that information is not transmitted on the core infrastructure at an excessively high frequency leading to draining of the battery power on the infrastructure. It also helps us to ensure that core infrastructure is updated only if there is change in the network environment. Therefore, in

Figure 3.3, nodes A and B transmit the information to neighboring core nodes. Node A transmits the message to B and D while B transmits the message to node A and C. Solid arrows represent the transmission of core update messages to the respective core nodes while dashed arrows show core update messages sent to neighboring core nodes. Core node A transmits a message containing target ids for P and W to core nodes B and D. Core node B transmits a message containing the target id for P to the core nodes A and C. By the time a message from A reaches node B, it has already a recent sequence number for P and it has shorter path to P cached as compared to the one contained in the message received from A. So node C updates the cache for the path to W, and forwards a core update message that contains the path to W to core node C. All the intermediate nodes keep on adding their energy level to the forwarded packet. Core node A receives an update message from B containing path information to P. But node A has already a path cached for W (A-Common Node- W) which is shorter in length to the path forwarded by B (A- common node- common node – B – W). Here we have eliminated the cycles in the source path before comparing the paths. We want to stress the point that since we take the weighted mean of energy level and hop count for determining a path length, even if the path cached by A has a smaller hop count, it will still be replaced by the path from B if the energy level for some node in the path cached by A is very low. Core node B forwards the message containing path to W and P. Core node C updates its cache. Before node C forwards this message to D, it receives the update message forwarded from D. Node C compares the path length of W in the message to that cached. It finds the cached path smaller for the same sequence number so it just drops this packet without any processing. Thus only the best routes are cached on core nodes. Upon receiving the core

update message, Core node D forwards the message only to C but not to A, since it has received the message from A.

All intermediate core nodes attach a timer that deletes the soft state associated with the target node upon its expiration. This indicates the unavailability of the target node. Referring to Figure 3.3, node C can no longer sense W if it does not receive an update message for target W before the timer expiration. Otherwise a new sequence number must have been forwarded for W on the core infrastructure. So C deletes the cache entry for the target W. We therefore do not need to transmit any cache clear message to clear cache entries in the core nodes. This saves energy by avoiding message flooding over the core infrastructure.

Choosing an update period is a design issue for our architecture. Choosing a large period will make the system store stale information about the target since by the time a new update message is generated, the target might have already moved away from the sensing node reach. On the other hand choosing a small period will add a lot of overhead to the update mechanism, as core nodes will be updated a number of times for the same target even though the target stays in the same core domain over the entire period. So we have chosen a simple model to select the time period for updating the core, as described next.

Upper Bound on Update Timers:

Assuming distance between neighboring core nodes = r meters

Maximum distance a target has to travel to enter another core nodes domain = 2r meters.

Minimum distance a target has to travel to enter another core node domain = 0 meters.

Average speed of target = v m/s.

Average time taken for the target to move into the domain of another core node = 2r/ v.

3.3 Query Forwarding

The goal of query forwarding through a virtual backbone infrastructure is to maximize network lifetime. This will help maximize energy efficiency. When a sink requires data, it sends a data query to its respective core node. The core node forwards the query to the upstream core node that provides the shortest path (taking energy usage into account) to the target using cached source route. The query is forwarded until it reaches the core node that announced the existence of data the sink is interested in. To stop the query from being forwarded indefinitely, we can take two approaches:

1) The first approach is to include a field similar to TTL in IP header [20]. The sink or a core node can specify the TTL field in a query packet. Each core node receiving the packet will decrement the TTL field value by one before forwarding it to the next upstream core node. When the field value reaches zero, the query packet will be discarded by the virtual backbone formed by the core nodes.

2) Since we are using source routing approach to forward the query, in the second approach we do not need any counter field since the source path will limit path length for the query to be forwarded.

We are using the second approach for limiting the path length to which the query will be forwarded. During the core update phase, all core nodes calculate the minimum length to the target (taking energy usage into account) and store this path in a cache for source routes, this path will be used to forward the query to the target core node. The query is forwarded using source routing until it reaches the core node that announced the existence of data the sink is interested in.

The strong point of core-based query forwarding is that the query need not be flooded even in a single core domain. A query will be handled directly by the neighbors sensing the target. This approach is much more efficient than the case where the query has to be broadcast to the entire sensor field. This approach will be even more efficient than the case where a query will result in flooding of at least a single cell area. This approach will also be very effective for query aggregation. When a core node receives the same query from different sinks, the core node drops duplicate query messages and forwards only one of the duplicate queries to upstream nodes. This approach is similar to the one used in TTDD, but we remove the formation of any cycles in data forwarding.

To use this query aggregation, we need a way to keep track of upstream nodes to which we need to forward the data. To achieve this, we just keep an index of the nodes from which query for a particular target is received. When an entry already exists for a target at a core node, we just drop the query at that point and keep an index of the upstream node from which the query came. Thus when data arrives, we just forward the data to all upstream nodes that sent a query. In other approaches like TTDD, cycles are formed in data forwarding since the same query could have reached a node from multiple

directions. However in our approach since a query is forwarded using source routing, it can only reach core nodes from one upstream node, thus eradicating any chances for the formation of cycles. Query aggregation is illustrated in Figure 3.4.

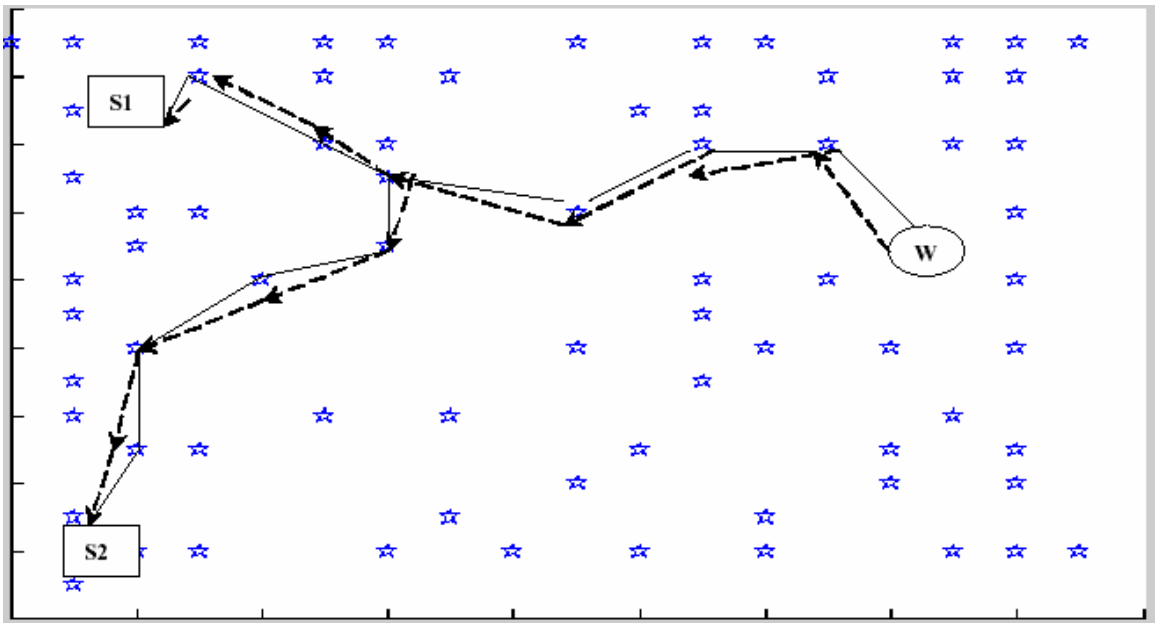

Figure 3.4: Query Aggregation at core nodes.

Here we see that when a core node receives queries for the same target from both S1 and S2 sinks, it drops the duplicate query and maintains an index for the node that forwarded the query. It just forwards one query message to downstream core nodes. Since each node maintains an index of upstream core nodes, a node does not need to maintain the source route for the data to be forwarded. In the case depicted in Figure 3.4, when data arrives for the target, the core node sends a copy of the data to both upstream nodes.

When a query message traverses the path to the target, timers are initialized at the core nodes in the path. A timer expires at a value that is preset by the administrator. This expiration time has to be higher than the rate at which the target produces data. The sink keeps on sending update messages to its respective core node for the query it initiated. The time period of sending update messages is smaller than the expiration time of the timers attached to the core nodes for the corresponding query. TTDD uses update messages to keep the respective query alive on the path on which the query is forwarded. If core nodes do not receive explicit update messages before the expiration of the timer, they stop forwarding the data to the sink. This approach eradicates any need for an explicit kill signal by the sink to the sensor node when the sink looses interest in the target's data. This approach is very scalable and efficient in the case where multiple paths exist from the target to the sink. But in our case there is a unique well-defined path between the sink and the target on which a query gets forwarded. Therefore, sending an explicit kill signal along that path will be much more energy-efficient as compared to the periodic update message. Therefore we have modified this approach in order to keep the query alive along the query-forwarding path. We removed the timer associated with the update messages. Now the update messages generated by the sink are handled by the sink's core node. Whenever the sink's core node receives an update message it resets a timer as described earlier. If the timer expires, it means either the sink is not interested in the target anymore or the sink has moved out of range from the core node. In this case, the core node sends an explicit kill message to the core nodes on the query-forwarding path. The kill message contains a unique id to identify the query message associated with it. Upon receiving the kill message, the core nodes in the path set up a timer with the

expiration period as discussed above. In case the sink moves to another core node's domain, the sink is asked to register itself with the new core node upon receiving any update message from the sink. Once the sink is registered with the new core node, a new query is generated by the core node on the shortest path to the target. If any core node on the previous path receives this message, it sends an update signal to all upstream nodes to remove the timer associated with the query. This update message is forwarded until it reaches the target's core node. The core node handling the query, updates the state associated with the downstream core node. If the timer expires, the nodes remove the downlink core node states for the query.

3.4 Data Dissemination

Once the core is formed, core nodes communicate through a modified dynamic source routing (DSR) protocol [16]. We have modified the DSR to suit our architecture.

Once a core node receives a query for an object sensed in its coverage area or that of its neighboring nodes, it forwards the query to the nodes that sensed the object and proactively forwards the information to the core node. The nodes sensing the target increase the frequency of data forwarding to the core node. The core node starts forwarding the data on the reverse path from which it received the query. The data is forwarded on the reverse path until the data reaches the core node to which the sink is registered. In case multiple sink nodes query for the same target, the core node will forward the data to all neighboring core nodes from which it received aggregated query messages.

The paths taken by packets to reach the sink can be suboptimal as compared to the case where direct broadcast is employed for query dissemination, as in the case of Directed Diffusion. In the case of a direct broadcast, a query is broadcasted to the entire sensor network area and the shortest path to the sink is selected for data forwarding. However with our approach the path taken can be *sqrt*(2) times the optimal path as is the case for TTDD. For example, Figure 3.5 shows the path taken by a query message from sink S to target T. Solid arrow shows the path for query flooding and dashed one for core based query forwarding. In case of query flooding, query follows a straight path from S to T. When core based architecture is employed, query is forwarded through neighboring core nodes thus query will be forwarded along the sides of right angle triangle with direct path as the hypotenuse.

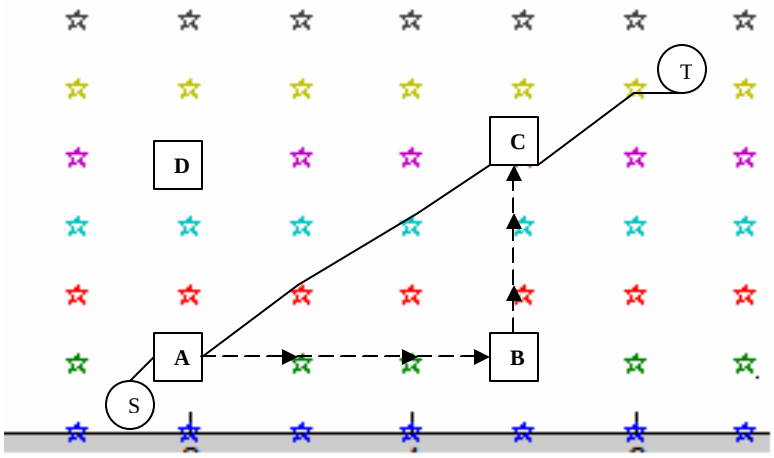

Figure 3.5 Path Lengths for Query Forwarding

3.5 Core Maintenance

To increase the robustness of the core infrastructure, we employ another node as the replica of each core node. We call this *node cluster backup node*. The core node periodically replicates its state to the backup node. When a new core node is elected in the network, the recent core node remains connected to the core until the newly elected node has resumed the routing responsibility. The energy resource manager schedules re-election of core nodes. The current core node and the backup nodes will have minimum probability of being re-elected as core nodes. This ensures that energy is drained uniformly throughout the cluster. The rotation performed ensures that the nodes are elected as core nodes in a weighted round-robin fashion. We therefore introduce five new packets in source routing, as described next:

1. BACKUP_BEACON: This is periodically sent by the core node to backup a node for state update. This is also used by a core node to update newly elected core nodes about its neighboring core nodes.
2. BACKUP_ACK: Sent in reply to the BACKUP_BEACON to ensure that the state has been updated and the backup node is alive.
3. JOIN_NODE: This is used by the newly elected core node to notify the neighboring nodes that a new core node will join the network. When neighboring nodes get the message, they create an alias of all the paths containing the core node that sent the message. All instances of the current node in source routes are replaced by the newly elected core node.

4. WELCOME: The neighboring nodes that receive a JOIN_NODE message transmit a WELCOME message to the newly elected core node.
5. KILL_NODE: The newly elected core node waits for the WELCOME message from all the neighboring nodes, and then it sends a KILL_NODE message to all the neighboring nodes. The neighbors delete all the entries in the routing table containing the obsolete core node. Then, this message is forwarded in the network. All core nodes update their routing tables to substitute entries of the obsolete core node with the newly elected one. The neighboring nodes keep the mapping between the newly elected core node and the obsolete core node, until timer expiration, directing traffic through the newly elected core node if the route contains the obsolete core node.

When the obsolete core node receives the KILL_NODE message, it relieves itself from its routing responsibilities and joins the group of sensor nodes in the cluster.

The modifications in the source routing hide the dynamics of the network from the routing protocol. Source Routing is very efficient in an ad hoc environment if the topology of the network is stable. With increasing dynamics of the network, energy efficiency degrades. This is due to the fact that with increased link failures, there will be a need for a lot of route discovery in the network. In our scenario, sensor nodes are static, but dynamics in the network are generated because of the rotation of core nodes. By ensuring that a newly elected core node takes the place of the obsolete core node before the node relieves itself, we hide the network dynamics from the protocol. Since all nodes in the core network are updated about the creation of a new core node,

no link failures will be faced. Before all nodes are updated, the distant core nodes may use the obsolete path containing the obsolete core node. However, this initial transition is masked by the neighbors of the newly created core node. The mapping stored at these nodes is used for this purpose.

One major challenge of our design is to handle failure of a core node. In this case the backup node will contain the obsolete state. But we assume that the network topology will remain stable if the period for BACKUP_BEACON is optimal. In this case, the backup node will just initiate a JOIN_NODE to the neighbors. Thus, a backup node enters the core network upon failure of its corresponding core node.

We provide support for reliability by using hop-by-hop recovery of the data as proposed in PSFQ. Thus, instead of flooding the entire network with data, only the target cluster will be flooded with data. On the core network, we provide hop-by-hop recovery. Since data will be transmitted only to neighboring nodes, an end-to-end approach for reliable transport is efficient to apply here. Since the backup node has the same state as the core node, upon failure of the core node, it can recover very efficiently by asking for the holes in the data received.

For the local cluster, we are using 802.11b DCF for data transmission. Our emphasis is on core-level reliability to the applications since PSFQ [7] argues that with a large number of hops, the success rate of the end–to-end system decreases exponentially. But with the core network, we have reduced the number of hops. Instead of going for complex protocols, we can achieve the same degree of reliability with traditional protocols as well.

Chapter 4

Simulation Results and Conclusions

4.1 Initial Performance Results

We are using a manually configured sensor network environment for all simulations. This configuration is suitable when we have full control of the environment we are planning to sense. However our algorithms are not tied to the sensors configurations. The algorithms can work without any need of extensions in any environment. So even if random distribution of sensors is inevitable, our algorithm will work perfectly fine without any modifications. In order to do so, we have ensured that the sensor nodes do not use any information being used to manually configure the nodes in the sensing environment for correct operation.

Initial simulation results are gathered with only one target node and one sink node. The target node and sink nodes are configured to be moving at a maximum speed of 5m/s, with a pause time of 0.10 seconds. The sink and target nodes move over the entire area covered by the network topology shown in figure 1.2.

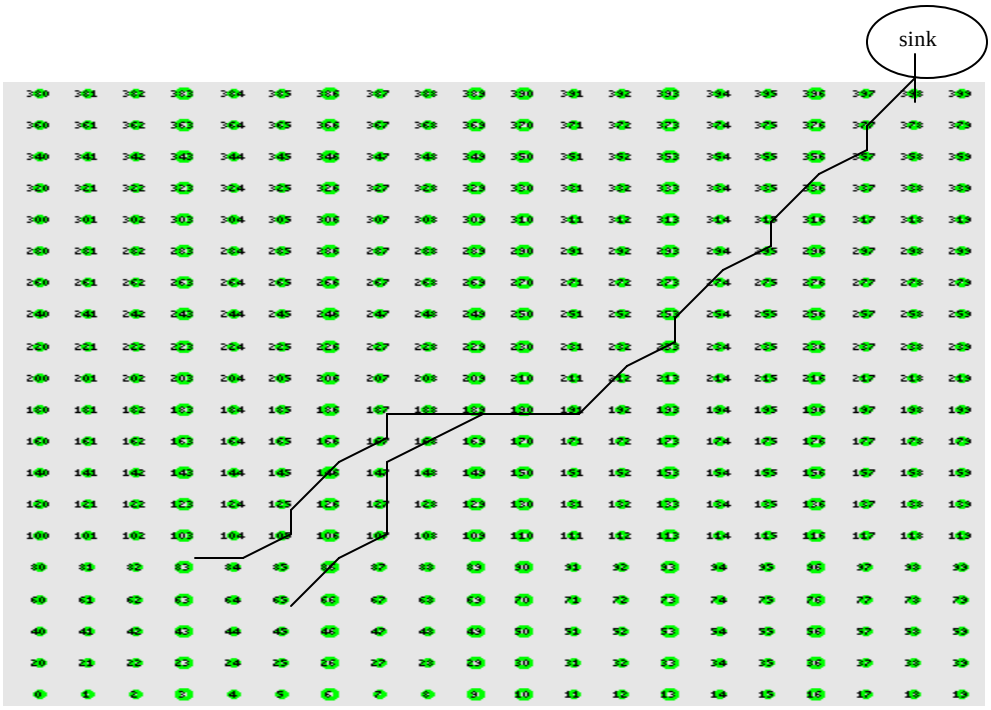

Figure 4.1: Basic Data Forwarding to the Sink.

Over that time we collect the sum and standard deviation of the remaining energy out of the initial 10 Joules assigned to all the. Then we collect the same set of data for the network topology but using a very simple clustering algorithm as described below. We fix the core sensor nodes at the start of the simulation. For example, Figure 4.1 shows a basic approach of data forwarding where all the nodes directly forward data to the sink. In Figure 4.2, node number 21 is elected as the core node for nodes 0, 1, 2, 20, 22, 40, 41 and 43. Now instead of engaging into route discovery and forwarding data directly to the sink, all nodes send their data packets to their respective core nodes. The core nodes then perform route discovery to the sink and forward the data packets to the sink as shown in Figure 4.2.

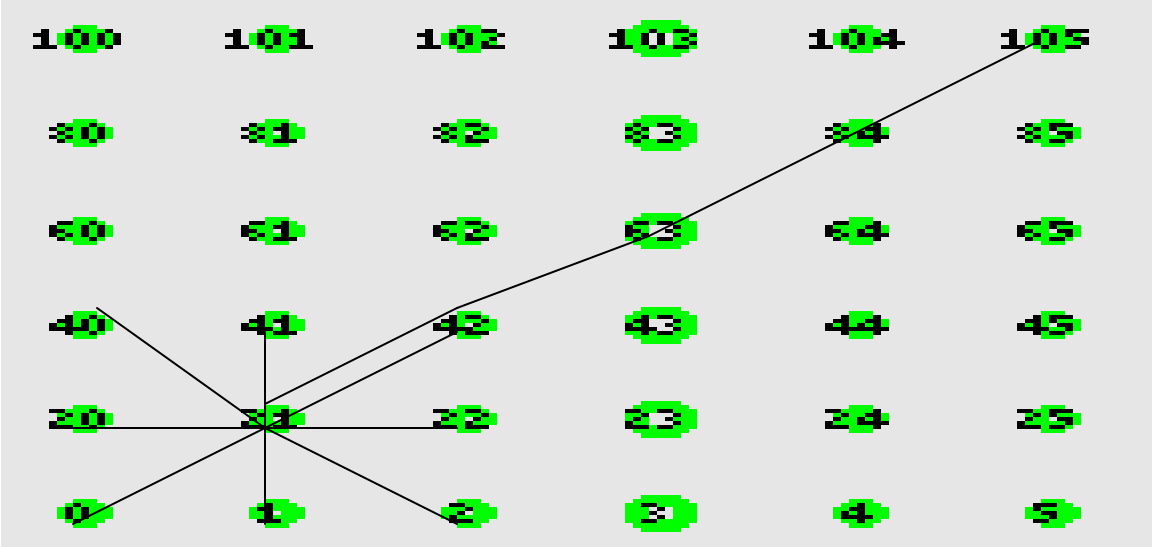

Figure 4.2: Data Forwarding through Core Nodes.

The results obtained are shown in Figure 4.3 and Figure 4.4. Figure 4.3 shows the total energy remaining for all nodes in the network topology and Figure 4.4 shows the standard deviation for the total energy of all nodes in the network topology.

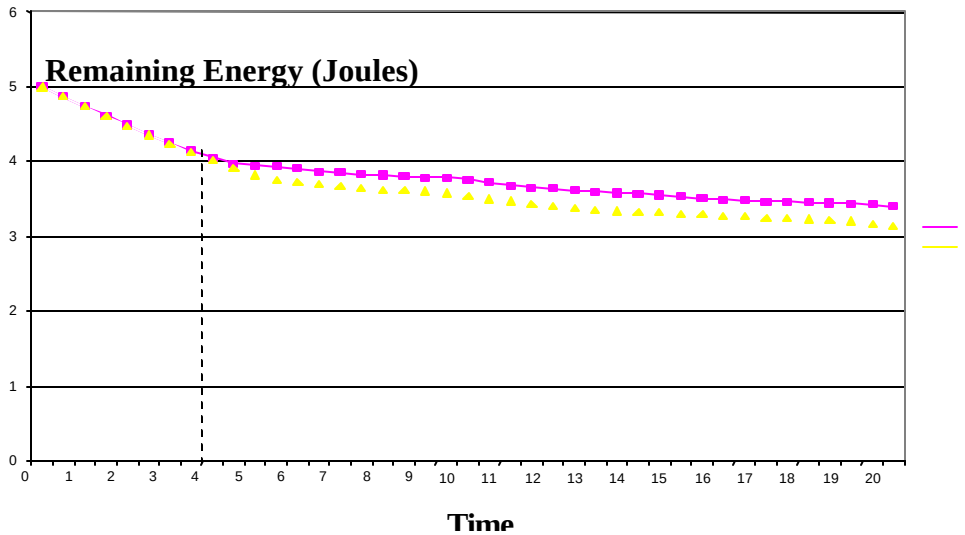

Figure 4.3: Mean of Remaining energy per node.

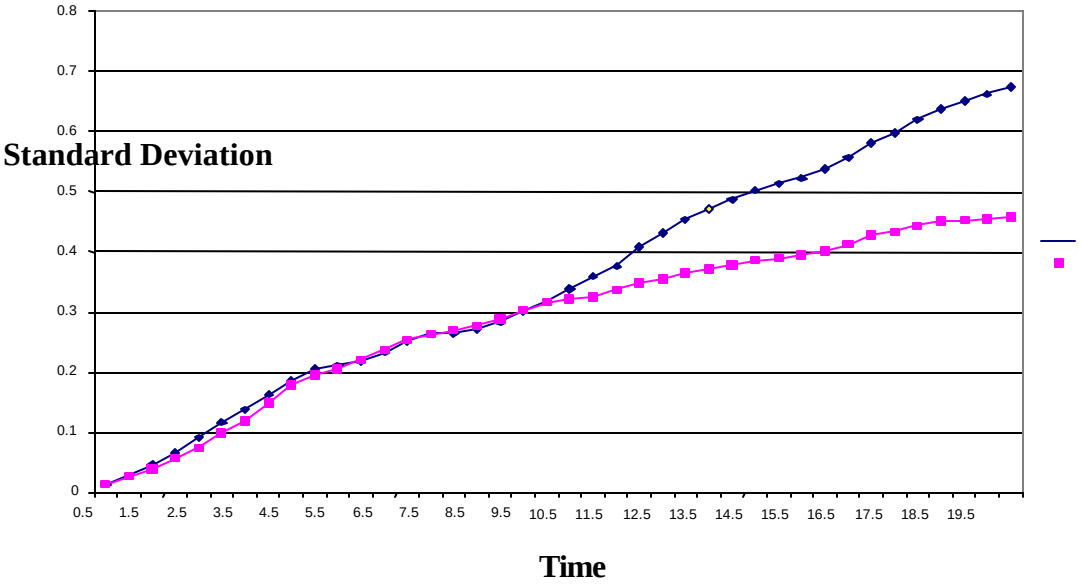

Figure 4.4, Standard Deviation of Remaining Energy per node

The results obtained were the motivation behind our proposed line of research. As can be seen in Figure 4.3, the average energy remaining per node is higher for our approach than the basic approach. We start with each node having 5 joules of energy. As time progresses, the basic approach spends more energy than the basic clustering approach. As time progresses the difference between the two plots keeps increasing. This confirms our intuition that, over a long period of time, a clustering algorithm can prove to be much more energy-efficient than basic shortest path routing algorithms. All optimizations proposed for shortest path routing algorithms can then be applied to the cluster-based approach for added energy efficiency. We can also see that the remaining energy per node decrease at higher rate for the basic shortest path approach as compared to the clustering approach. This shows that a clustering-based approach can be used to increase the lifetime of individual sensor nodes.

Figure 4.4 plots the standard deviation for the remaining energy per sensor node. The graph shows that standard deviation remains lower for a cluster-based approach as compared to the basic approach. We also see that the standard deviation increases over time, but it increases at a lower rate for the cluster-based approach as compared to the basic approach. Also the standard deviation for the cluster-based approach flattens out as time progresses. This confirms our initial assumption that with a cluster-based approach all sensor nodes spend energy more uniformly. A cluster-based approach can therefore be efficient in extending the network lifetime. Moreover, since we are ensuring that all nodes spend their energy more uniformly, the point in time when network partitioning would occur, due to nodes exhausting their batteries, can be extended. The results obtained are due to the fact that 8 out of every 9 nodes are forwarding their packets to

their respective core nodes. Therefore, out of every 9 nodes, 8 nodes are following a similar transmission behavior, spending energy at same rate as a result. Thus (8/9)*100 = 88.89% of the nodes will have a similar transmission pattern resulting into mostly the same energy spending pattern. This explains the better standard deviation obtained by the clustering algorithm as compared to the basic approach.

4.2 Performance Evaluation of Architecture

Out of 400 nodes, 81 nodes are elected as core nodes in the simulation. This can be made more efficient at the expense of a more complicated design, as will be discussed later.

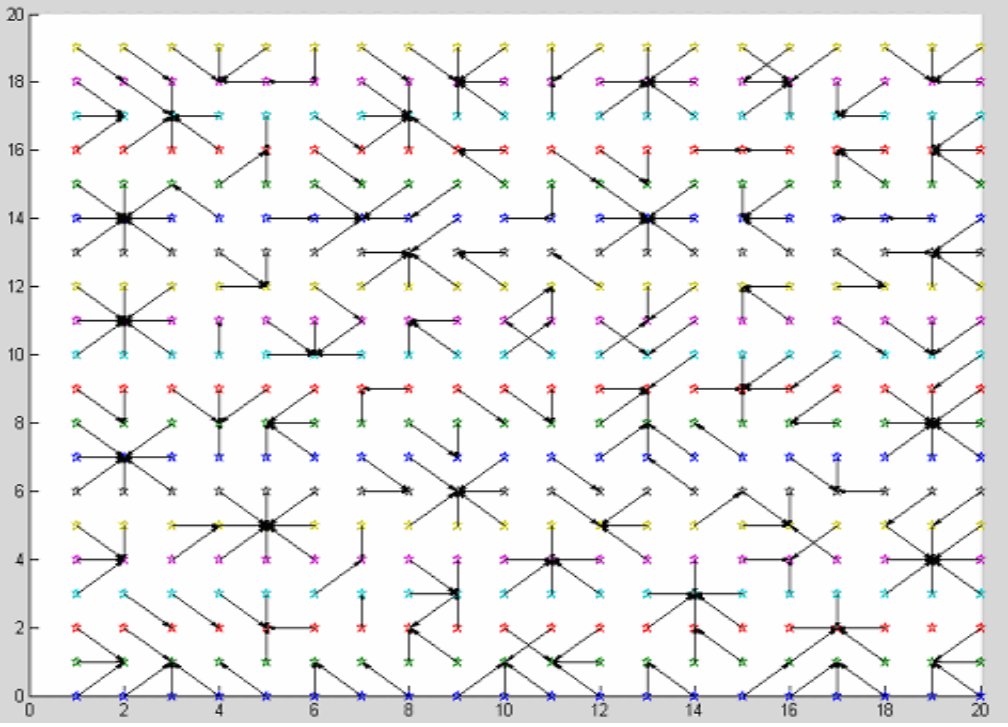

Figure 4.5: Elected Core Nodes.

Figure 4.5 shows the result of core formation.

Here we see that some of the nodes elected as core nodes have chosen other nodes as their core nodes. This is an outcome of the approximate solution we are using to solve the NP-hard minimum dominating set problem. We can improve this with some extra message exchanges but it will lead to extra overhead for core formation. The overall architecture will function perfectly fine with some extra core nodes included.

We want to stress the advantage of using the degree of neighboring nodes in selection of core nodes. We see that the algorithm does not favor electing nodes on the boundary of the sensor field as core nodes. This is a direct result of the fact that nodes having a greater number of neighboring nodes are favored to serve as core nodes. If a node on the boundary is elected as a core node, it will serve a smaller number of neighbors than ones that are located more towards the center of the sensing field.

Figure 4.6a and 4.6b shows the results of core node election with one- and two-hop transmission of core formation packets. We see a considerable decrease in the number of core nodes being elected as we expected. But this also is an indication of an increase in the network load on each of the core nodes. The other factor we want to stress is the effect of the network load on the distributed algorithm. The results show that nodes that are one-hop (or two-hop, depending upon the number of hops used for packets transmission) away from the boundary are preferred over other nodes for being elected as core nodes. The simulation results show that this is due to the lower network contention towards the boundaries as compared to the central sensing zone. We saw a greater number of packets dropped by the nodes towards the center as compared to those on the

boundary. After neighbor discovery, the nodes towards the boundary update the neighboring nodes about their own degree and degree of dominance as discussed in section 3.1. This information is transmitted by the nodes near the boundary in a low contention environment thus the variation in the time of information retrieval about the neighbor's dominance and degree is low. Therefore, core nodes are selected uniformly at one-hop boundary. While towards the center, due to high contention, there is high variance in the time for nodes to receive broadcast information. Moreover, due to using 802.11b as the MAC layer protocol, once a node gets a chance to transmit a packet, it captures the channel until transmission is completed. As a result, due to this variation in information retrieval, once a core node announces its status, all neighboring nodes waiting for complete information select it as their respective core node. This seeming unfairness introduced in the algorithm is due to network contention.

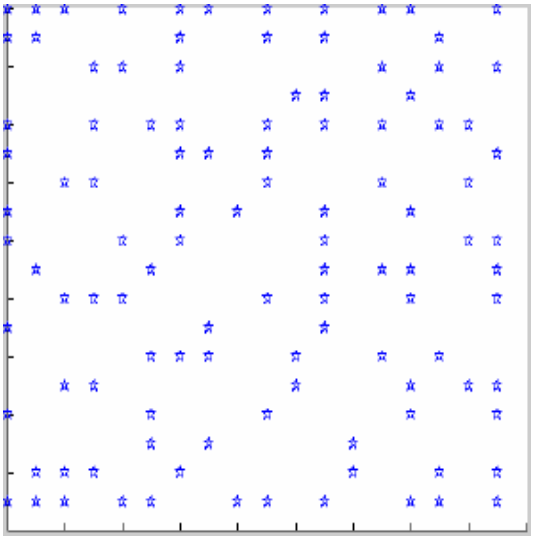

Figure 4.6a: Elected Core Nodes with One-Hop Packet Transmission.

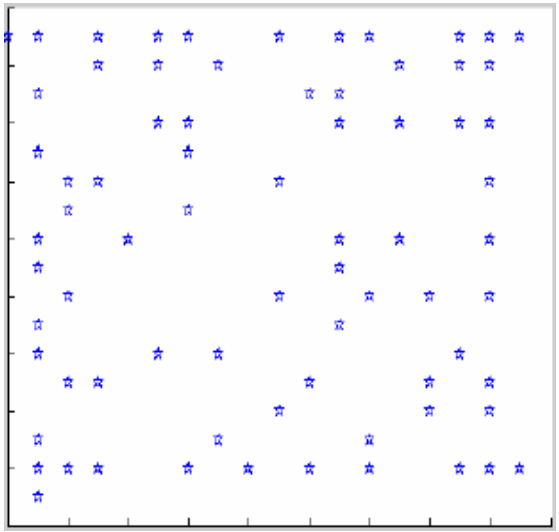

Figure 4.6b: Elected Core Nodes with Two-Hop Packet Transmission.

Figure 4.7 shows the result of core formation in terms of the remaining average energy per node versus transmission power of the nodes. Figure 4.7 confirms the common thesis arrived at by existing work on topology control of multi-hop wireless networks using transmission power control. These works prove that for optimal energy usage, the transmission range used by nodes should be kept to a minimum to keep the network connected [17]. This can be attributed to the fact that transmission energy is a function of transmission power per hop and the number of hops data is transferred. When we increase the transmission range, the number of hops decreases linearly while the power per hop increases exponentially with an exponent of 2 to 4 [18]. As a result, the energy consumed in a network is minimized with minimum transmission range. This can be seen from Figure 4.7. It shows a plot of the remaining energy for different values of the transmission range. It clearly supports the minimum transmission power thesis. We see that the average remaining energy decreases as we increase the transmission power. This supports using the minimum transmission range combined with multi-hop forwarding for core formation instead of using extended transmission range. It is one of the design parameters, since intuitively it seems that extending the transmission range will result in the transmission of a much smaller number of packets – a larger number of nodes can receive broadcast packets. At the same time, using a decreased transmission range will increase network load since every packet has to be forwarded through a large number of hops. This may adversely affect overall network throughput. But this can be offset by the fact that spatial reuse of the network increases quadratically with decreasing transmission range [18]. This is so because the inhibition range of a transmission is proportional to the square of the transmission range. Thus any linear increase in the

network load due to multi-hop data transmission can be easily offset by the spatial re-use factor.

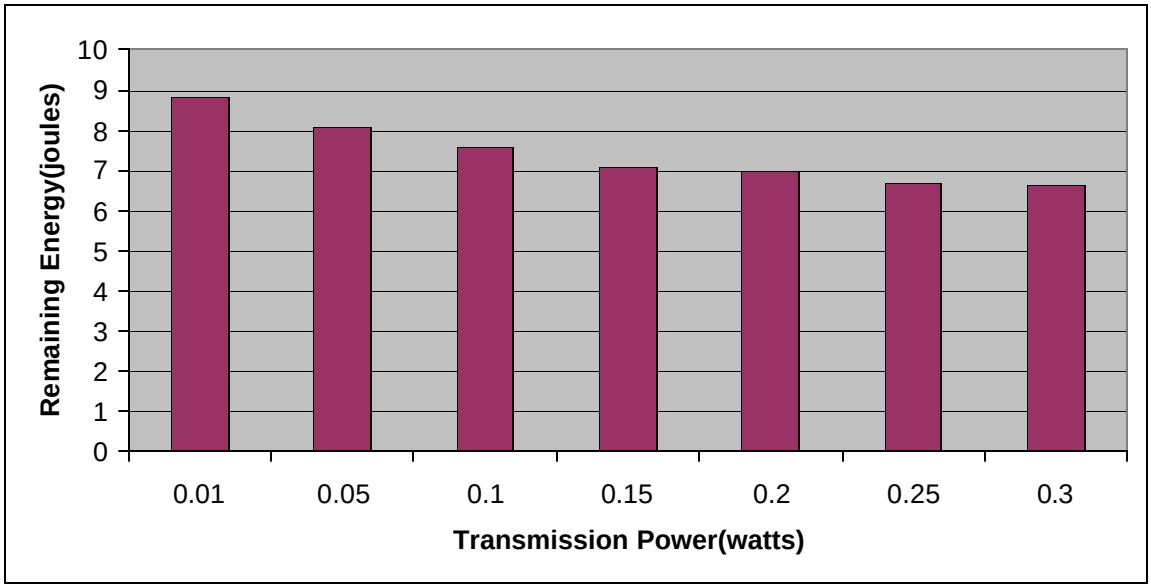

Figure 4.7: Remaining Energy (joules) vs Transmission power.

The other major factor to be considered for choosing between extended transmission range and a minimum transmission range with multi-hop forwarding is the simplicity of the protocol. We definitely do want to keep the network protocol as simple as possible for a resource constrained system. So we studied the performance of the extended transmission range approach for core formation. We were expecting a gradual decrease in the number of core nodes elected with the increase in transmission range. Intuitively, the increase in transmission power should lead to a higher degree of

clustering resulting in a lower number of core nodes to provide coverage for the entire network. Figure 4.8 shows the results we obtained. The result obtained was contradictory to what we expected. We saw anomalies in the result. At the transmission power of 0.1 Watt, the numbers of core nodes were higher than at 0.05 Watt. On analysis of the packet exchanges, we found that at the 0.01 Watt power level, only the side nodes were able to hear the broadcast messages. While on increasing the level to 0.05 Watt, diagonally opposite nodes also came into transmission range of each other. This led to a decrease of the number of core nodes. The increase in the number of core nodes at 0.1 Watt can be attributed to the fact that increasing the transmission power twofold did not bring next hop nodes in communication range. Increasing power level only resulted in a higher number of collisions in the network without doing anything fruitful. This led to decreasing the effective transmission range of increased transmission power. This gave us a clear insight that using an extended transmission range approach will require an accurate model of the radio interface for efficient operation of the core. This is an extra overhead on the core layer formation. With multi-hop transmission we are sure that increasing the number of hops will certainly bring the next hop node in the transmission range of a node. As a result, we use the minimum transmission range with multi-hop forwarding in our implementation.

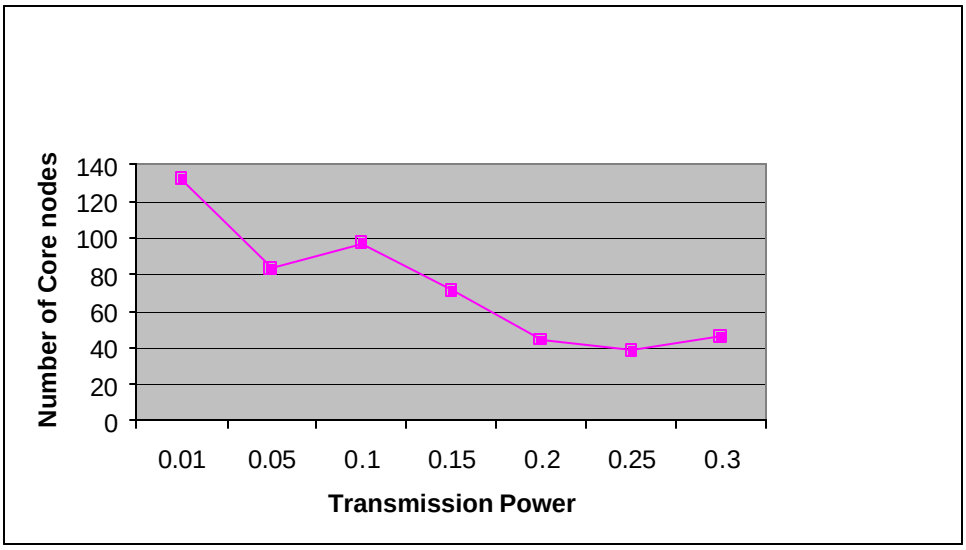

Figure 4.8: Number of Core Nodes vs Transmission Power.

Routing Layer: The link layer in ns2 calls ARP only once for each packet it receives. If a second packet arrives to the same next hop neighbor before the ARP response is received, the first packet is dropped because there can be at most one packet for the same neighbor in the ARP table. This may cause some performance penalties. For example, an agent can have many data packets from multiple sources for the same sink, if the sink is interested in data from multiple targets. In the case of heavy traffic, all packets can get dropped. To address this issue, a queue is maintained in the routing agent. Each packet from higher layers is queued first. After a packet is sent to the link layer, a timer is set to expire after an ARP_WAIT_TIME that is randomly distributed between 0.01s and 0.02s. When the timer expires, it checks the link layer's ARP table to see if the previous packet has been sent out. If so, it removes form the queue all packets with the same next hop neighbor and sends them out. Otherwise, it sends the packet to the link layer again. A

packet is dropped and the next packet is sent out after MAX_RESEND_COUNT number of retries.

Chapter 5

Future Directions

At present we are just using the general weights to calculate the shortest path for data transmission and query forwarding. A more accurate mathematical model can be used instead for route computation. Simulation plots of energy consumed for different values of weights can be used to obtain suitable weights for evaluating the minimum path length.

The query update algorithm can be further improved. Using our proposed update mechanism, even if there is a temporary improvement or degradation over some path, this information will be transmitted every time over the network. However, for temporary changes, the network state changes back to the original one and, once more, another message has to be propagated in order to change the soft-states. An approach similar to CEDAR can be used to handle this problem; i.e. An approach of update slowly, kill quickly can be very efficient for this purpose [14]. The update message will be forwarded with a time delay. Thus if the network again reverts back to its original state, a kill signal traveling faster than the update message will be sent to stop updating the core soft state and revert the core's state to the previous one.

Computation of query forwarding timers gives an upper bound on the update message timer. However, due to network latency, the timer for message updating should be chosen in such a manner that there is sufficient time to transmit the message to the core network.

If we do not ensure this, then remote core nodes will be storing some obsolete states due to the network latency introduced by the update messages traversing the network. Assuming network latency is 'L', then the update period has a lower bound of 'L'. However, we should keep the update period larger than this since, otherwise, there will always be incoherency in soft states except at times that are multiples of 'L'. Therefore, in order to keep the network stable we should keep the update period larger than 'L'. To compute 'L', TCP's retransmission timeout calculation can be modified for our architecture.

An additional area for future work is sink mobility. Sink mobility must be handled to provide seamless connectivity to the sink. An algorithm must therefore be developed to provide a soft handover to the sink. Once the data reaches the sink's core node, local data forwarding algorithm should be employed by the core node to forward data to the sink.

With growing interest in applications that demand certain end to end performance guarantees, the introduction of imaging and video sensors has posed research challenges for providing QoS services in sensor networks. This is necessary to ensure efficient usage of sensor resources and effective access to the gathered measurements [21]. These applications require the routing protocol to provide least-cost, delay-constrained path in terms of link cost that captures the nodes remaining energy, error rate and other communication parameters. The routing protocol should also maximize throughput for non real-time data using best effort service. QoS in wireless network is an active research area and is still emerging. Our core-based architecture is capable of supporting basic QoS

since it stores the shortest data paths during core updating. Energy usage has been identified as the main component of QoS in sensor networks and our architecture includes it in computing shortest routes. But we need to make improvement to our architecture for providing QoS in the case where different applications or users have conflicting requirements for QoS parameters. Our architecture uses single combination of QoS parameters over the entire sensor network.

We did not develop our architecture with security as a goal. But with insecure wireless communications, limited node capabilities and possible insider threats, security, intrusion tolerance and high availability have become critical issues in sensor network deployment. Alhough core maintenance provides certain degree of high availability in our architecture, security and intrusion detection has to be included for most applications. However common authentication schemes are not applicable in sensor networks since a central authentication authority is hard to be deployed in the sensing environment. As a result, a distributed authentication facility must be used in the network. A cluster-based security scheme can be used in our architecture for providing distributed authentication [22]. The approach requires formation of clusters with one special head node in each cluster. These head nodes execute administrative functions and hold shares of the network keys used in authentication. Nodes can join the network only as guests. They can become full members only after a head node has authenticated them. In our architecture, core nodes are fully capable of performing administrative functions and thus integration of the algorithm should not introduce substantial overhead.

Chapter 6

Conclusion

Wireless sensor networks have been an active area of research in recent years. Sensor nodes are resource constrained in a number of ways, but the main factor is energy since it decides the lifetime of the nodes. In this work we described the design and implementation of a core-based architecture for energy-efficient data dissemination, data aggregation, query forwarding and reliable transport. The core update phase proactively floods target information over the network so that each sink's query broadcast is limited to a local cluster. Core infrastructure also provides efficient data and query aggregation without extra message exchange overhead. Data is forwarded on the reverse path on which a query is forwarded to the target core node. Our simulation results have confirmed the efficiency of the proposed design. Energy efficiency achieved by our design demonstrates the effectiveness of a virtual routing infrastructure that is based on clustering in stationary wireless sensor networks. We have also discussed here the architecture could support basic QoS and security services for applications.

Appendix

A. Shortest Path Calculation

QoS measures can be classified into additive (e.g., delay, number of hops) and non-additive (e.g., energy, available bandwidth, policy flags). In the case of an additive measure, the QoS value of a path is equal to the sum of weights of the corresponding links along the path. For a non-additive measure, QoS of a path is the minimum (or maximum) of the link weight along that path. In general, non-additive measures can easily be dealt with by pruning all the links that do not satisfy required QoS constraints. Additive measures require more complex computation. Therefore, for finding the shortest path we mainly have to consider additive measures.

In our architecture, the remaining energy is a non-additive measure and the number of hops is an additive measure. We define the shortest path length as

Path Length =

$$\frac{a}{\min(remaining\_energy\_on\_a\_path)} + \frac{b}{number\_of\_hops\_on\_a\_path}$$

The scaling factors a and ß give us full control to tune the path length with the remaining energy and number of hops as parameters.

B. Pre-Defined Energy Threshold for Core Election

To keep the architecture simple and scalable, the node election process is periodic. Therefore, nodes periodically collaborate to elect a new core for the local cluster. Core node and backup cluster nodes do not get a chance to participate in the next round of core election. However, to minimize the probability of failure of a core node, nodes that are already below the pre-specified threshold of remaining energy are also barred from being elected as core nodes. We used a random chosen value as the threshold. More accurate values can help maximize network partitioning time. For evaluation of accurate values, we need to gather statistical data related to energy consumption of a core node. Based on the mean and standard deviation of energy consumption, a node can derive from its remaining energy level whether it can perform as a core node before the next election phase. If it cannot, the node will abstain from participating in core election.

References

[1] Clare, Pottie, and Agre. Self-Organizing Distributed Sensor Networks. In *SPIE Conference on Unattended GroundSensor Technologies and Applications*, April 1999, pp. 229–237.

[2] Wang, Iyengar and Chakrabarty. Multisensor data fusion in distributed sensor networks using mobile agents. In *Proceedings of International Conference on Information Fusion,* August 2001, pp. 11-16.

[3] Intanagonwiwat, Govindan and Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *ACM International Conference on Mobile Computing and Networking (MOBICOM 2000),* August 2000, pp. 56-67.

[4] W. Heinzelman, J. Kulik and H. Balakrishnan. Adaptive Protocols for Information Dissemination in Wireless Sensor Networks. In *Proceedings of 5th ACM/IEEE Mobicom Conference (MobiCom '99),* August 1999, pp. 174-185.

[5] Ni, S.Y., Tseng, Y.C., Chen, Y.S., Sheu, J.P. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99),* August 1999, pages 152-162.

[6] Haiyun Luo, Fan Ye, Jerry Cheng, Songwu Lu, Lixia Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *Proceedings of the 8th annual international conference on Mobile computing and networking,* September 2002, pp. 148-159.

[7] C. Y. Wan, A. T. Campbell, and L. Krishnamurthy. PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks. In *Proceedings of the First ACM International*

Workshop on Wireless Sensor Networks and Applications (WSNA 2002), September 2002, pp. 1-11.

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

[9] Samuel R. Madden. Query Processing For Streaming Sensor Data. *Ph.D. Qualifying Exam Proposal,* May *2002.*

[10] Samuel Madden, Mehul Shah, Joseph M. Hellerstein, Vijayshankar Raman. Continuously Adaptive Continuous Queries over Streams. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data,* June 2002, pp. 49-60.

[11] Wendi Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-Efficient Communication Protocols for Wireless Microsensor Networks. In *Proceedings of Hawaiian International Conference. on Systems Science,* January 2000, pp. 295-314.

[12] http://standards.ieee.org/getieee802/802.11.html

[13] M. Bhardwaj, T. Garnett, and A. Chandrakasan. Upper Bounds on the Lifetime of Sensor Networks. In *IEEE International Conference on Communications,* vol. 3, June 2001, pp. 785-790.

[14] Raghupathy Sivakumar, Prasun Sinha, Vaduvur Bharghavan. CEDAR: A Core-Extraction Distributed Ad Hoc Routing algorithm. In *IEEE Journal on Selected Areas in Communications,* v 17, n 8, Aug. 1999, pp. 1454-1465.

[15] http://www.isi.edu/nsnam/ns/

[16] David B. Johnson, David A. Maltz, Josh Broch. DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks. In *Ad Hoc Networking, edited by Charles E. Perkins,* Chapter 5, Addison-Wesley, 2001, pp. 139-172.

[17] R. Ramanathan and R. R-Hain. Topology Control of Multihop Wireless Networks using Transmit Power Adjustment. In *Proceedings of IEEE INFOCOM*, 2000, pp. 404-413.

[18] David Eppstein. Finding the K Shortest Paths. *Tech Report 94-26*, May 31 1994.

[19] RFC 2205 - Resource Reservation Protocol (RSVP), http://www.ietf.org/rfc/rfc2205.txt, September 1997.

[20] RFC 791 – Internet Protocol, http://www.ietf.org/rfc/rfc0791.txt, September 1981.

[21] Dazhi Chen and Pramod K. Varshney. QoS Support in Wireless Sensor Networks: A Survey. *Technical Report, http://web.syr.edu/~dchen02/research.htm*

[22] M. Bechler, H.J. Hof, D. Kraft, F. Pählke, L. Wolf. A Cluster-Based Security Architecture for Ad Hoc Networks. In *IEEE Infocomm 2004*.