Design, Implementation, and Evaluation of Node Placement And Data Reduction Algorithms For Large Scale Wireless Networks

A Thesis
Presented to
The Academic Faculty

By

Hardik Mehta

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
November 2003

Design, Implementation, and Evaluation of Node Placement And Data Reduction Algorithms For Large Scale Wireless Networks

Approved by:

[illegible] 11/13/03

Professor Douglas M Blough, Advisor

[illegible] Nov 13, 2003

Professor George F. Riley

[illegible] 11/13/03

Professor Henry L. Owen

Date Approved 11/13/03

ACKNOWLEDGEMENTS

I would like to express my gratitude to all those who have helped me complete this thesis. I am grateful the thesis committee for having approved my thesis proposal and granted me the permission to proceed with my research.

I am deeply indebted to my advisor Prof. Douglas Blough, whose guidance and support has led to the completion of this thesis. I am also thankful to Dheeraj Reddy and Scott Marelette for helping me with part of my thesis. I am obliged to Dr. Riley of the ECE, Georgia Tech, for all his help, suggestions and valuable hints in the research. I would like to thank all my colleagues in the Critical Systems Laboratory for their suggestions and support during my research and writing of this thesis.

I would especially like to thank my parents whose encouragement, love, and support enabled me to complete this work.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

LIST OF TABLES vii

LIST OF FIGURES viii

SUMMARY x

I. INTRODUCTION 1

II. BACKGROUND AND RELATED WORK 4

III. PROBLEM STATEMENT 9

IV. DESCRIPTION OF DATA REDUCTION PROTOCOL 12

4.1 Lowest Level Protocol 12

4.1.1 The Collector Node 13

4.1.2 The Leaf Nodes 14

4.1.3 The Intermediate Nodes 14

4.2 Higher Level Protocol 16

4.2.1 The n-level Collector........16

4.2.2 Intermediate Collector Nodes........19

4.2.3 Leaf Collector Nodes 21

V. Implementation of Node Placement Algorithms 23

5.1 Non-Clustered Distributions 23

5.1.1 Bounded Normal Distribution 23

5.1.2 Uniform Distribution 24

5.1.3 Radial Distribution 25

5.1.4 Poisson Distribution 26

5.2 Clustered Distributions 27

VI. Implementation of Data Collection Protocols 31

4.1 Data Reduction Protocol 31

4.2 TAG and Concast Protocols 33

VII. Analysis of the Results of Data Collection Algorithms 34

7.1 Performance Tests over Uniformly Distributed Networks 34

7.1.1 Number of Packets Required Per Collection 36

7.1.2 Number of Hops Required Per Collection 37

7.1.3 Amount of Time Required Per Collection 39

7.1.4 Number of Packets to Compute Min/Max 40

7.1.5 Number of Hops to Compute Min/Max 42

7.1.6 Amount of Time to Compute Min/Max 43

7.2 Performance Tests Over Bounded Normally Distributed Networks 44

7.2.1 Number of Packets Required Per Collection............45

7.2.2 Number of Hops Required Per Collection 46

7.2.3 Amount of Time Required Per Collection 47

7.2.4 Number of Packets to Compute Min/Max 48

7.2.5 Number of Hops to Compute Min/Max 49

7.2.6 Amount of Time to Compute Min/Max 50

7.3 Performance Tests over Different Radio Ranges.................51

7.3.1 Packets Required Per Min Operation 52

7.3.2 Hops Required Per Min Operation...........................52

7.3.3 Amount of Time Required Per Min Operation 53

7.4 Performance Tests with Loss of Nodes...............................55

VIII. CONCLUSION 58

8.1 Summary of Results..58

8.2 Future Work 60

REFERENCES 62

LIST OF TABLES

1 Timeout Values for TAG Protocol 56

LIST OF FIGURES

1. Bounded Normal Distribution 24
2. Uniform Distribution 25
3. Radial Distribution 26
4. Poisson Distribution 27
5. Uniform-Uniform Cluster Distribution 29
6. Uniform-BoundedNormal Cluster Distribution 30
7. Data Reduction Protocol over a Sample Network 32
8. Average Number of Packets per Collection in Uniform Distribution....36
9. Average Number of Hops per Collection in Uniform Distribution........37
10. Average Amount of Time per Collection in Uniform Distribution........39
11. Average Number of Packets Required to Compute Min/Max in Uniform Distribution 40
12. Average Number of Hops Required to Compute Min/Max in Uniform Distribution 42
13. Amount of Time Required to Compute Min/Max in Uniform Distribution 43
14. Average Number of Packets per Collection in Bounded Normal Distribution 45
15. Average Number of Hops per Collection in Bounded Normal Distribution 46

16. Average Amount of Time per Collection in Bounded Normal Distribution....47

17. Average Number of Packets Required to Compute Min/Max in Normal Distribution....48

18. Average Number of Packets Required to Compute Min/Max in Normal Distribution....49

19. Amount of Time Required to Compute Min/Max in Normal Distribution....50

20. Average # of Packets vs. Radio Range....52

21. Average # of Packets vs. Radio Range....53

22. Average Amount of Time vs. Radio Range....54

23. Average Percentage Error for Count with Loss of Nodes....56

24. Average Collection Latency for Count with Loss of Nodes....57

SUMMARY

This thesis develops a new hierarchical protocol for efficiently collecting data from nodes of a sensor network by performing in-route data reduction. The goal is to develop a protocol that requires small amount of processing and scales well for large networks. A network with three hierarchical levels is built and used to evaluate the protocol. The performance of the protocol is based on the number of packets, number of hops and time required to perform a collection over the network. Uniform, Poisson, and bounded normal random distributions are implemented and used for node placement for the sensor network.

The types of collection operations performed are sum, min, max over the sensor network. The protocol allows different types of collection at different levels in the network. The lowest level has a different protocol than higher levels. The performance of the protocol is compared with collection using no reductions, collection using Concast algorithm [7] and collection using TAG algorithm. [14]

The thesis is structured as follows:-The introduction section gives a brief summary of the need for efficient aggregation in large sensor networks and applications of the protocol. This is followed by a brief history of the work that has been done so far to solve this problem. The problem statement, description of protocol, and simulation details are explained in the following chapters. A detailed analysis of the results of performance tests is presented next, which is followed by the conclusion and future-work.

CHAPTER I

INTRODUCTION

Wireless sensor networks consist of multiple network nodes distributed over an area to measure the state of that area. Each node can contain some environmental sensing device for temperature, audio, video etc. Each node also has some limited computational power. Because of recent advances in sensors and wireless communication technology, the size of these sensor nodes is getting smaller and smaller while the number of sensors in a single network is increasing. The future applications of sensor networks will consist of thousands of low cost, low power autonomous nodes deployed over a region of interest.

One such application of wireless sensors is described in [6]. A set of sensor nodes were deployed on a small island off the coast of Maine to monitor habitat and behavior of seabirds. The sensors sent periodically collected live data to the web. Wireless sensor networks can be used in a wide variety of applications from sensing the temperature and pressure on surface of space shuttles to monitoring traffic flows on highways [8, 10, 11]. In [9], Kaiser talks about sensor networks consisting of millions of dust size (cubic millimeter) sensors deployed to gather information about an environment.

The common feature in all the applications described above is the need for sensors to combine their data to provide some useful information to the user. This leads to frequent many to one flows where data from many sensors is sent to a sink or collector point to be analyzed or sent to external application or user. Having each sensor transmit its data individually to the collector would be too expensive for power limited nodes. This would also cause too many collisions in a large size network and would put too much strain on the collector. There has to be a method to reduce the amount of data in route to the collector and to minimize the number of packets transmitted in the network. A good protocol would maximize efficiency by reducing data as much as possible, but would have to be low complexity due to limited computational power available at each node.

Recent research on data reduction in wireless sensor networks has focused on developing application-specific protocols. The focus of this research is to develop a general data reduction protocol that can be used with small modification by any application that requires data aggregation. The protocol is designed for scalability and requiring low amount of computation at each node and the collector. This study evaluates the performance of this protocol for random sensor networks with regards to number of packets and transmission hops required for data collection.

Definitions

- Sensor Node: Network nodes with some type of sensor (radio, temperature) attached to it. They usually have low amount of computational and transmission power, and usually run without direct user control.
- Wireless Sensor Network: Consists of a large number of sensor nodes communicating with each other to gather information about region of interest.
- Data Collection: Collecting data from sensor nodes at a sink (collector node) to measure some state of the system.

- Data Reduction: Combining data from multiple nodes in some meaningful way to reduce the amount of data sent to the collector.
- Spanning Routes: A set of multi-hop routes such that every node in a region is at least one of the hops in one of the routes in the set.
- Dynamic Source Routing (DSR): Source routing protocol developed by Johnson et. al. for ad hoc wireless networks. The source discovers the route to the destination node and includes this route in the transmitted packets [2].

CHAPTER II

BACKGROUND AND RELATED WORK

This project is a part of a larger ScalableSensorSim project being conducted by Dr. Riley and Dr. Blough [5]. The main goal of the larger project is to design and implement highly scalable simulation environment for wireless sensor networks. The larger project also involves implementation of protocols designed specifically for large-scale sensor networks that optimize data throughput and overall life of the network elements. The sensor nodes usually have very limited processing power and memory resources. Thousands of these types of devices can be dropped in an area to collect measurement data. Researchers at UC Berkely have been working on developing very small sensor elements called smart dust motes. They have developed TinyOS operating system to facilitate development using their small motes [13].

Since data collection is done periodically in these types of sensor networks, an efficient method to collect data from a large set of sensors is a major issue. The naive approach of having each sensor sending its data directly to the collector would create collisions and unnecessary traffic in the network. The collisions at the collector would increase as the size of the network increases. One potential solution to this data collection problem is described in [7]. Concast proposes a technique that could be described as reverse of multicast. Multiple senders are represented by single address and messages from this group are combined on route to the receiver. Concast provides an application-specific merge function to merge data from multiple senders into a single message for the collector. The collector creates a Concast Signaling Protocol (CSP) that specifies what sensors to collect from and how to merge the data. This CSP is then sent to all one hop neighbors of the collector. Each of these neighbors creates a flow state and upstream neighbor list. The collection request is broadcasted through the sensor network. When a node receives a Concast reply message from a sender, it looks for CSP for that message. If the node does not have a CSP, it sends out a request for merge specification to the receiver. Each node on route to receiver becomes parent node of the previous node on the route, and forwards the request to next node. This is done until request reaches a node that knows the CSP. A Concast tree is constructed in this manner. Nodes with more than one upstream neighbor can perform merge of data packets. For example, if the data being collected is the minimum of some measurement, the Concast node would not forward any data received that is larger than previously sent data. This reduces the number of messages to the collector. This algorithm does not provide significant benefits when data being collected, like sum of temperature from each sensor, cannot be reduced. Concast nodes can become bottlenecks, and no benefit is gained if packets arrive in reverse order from the way they are being reduced. Concast also requires each node to keep some state information about value previously forwarded and the list of upstream neighbors. This is undesirable for memory-constrained sensors nodes.

Directed diffusion is a method to query specific pieces of information from specified regions of the sensor network [15]. The collector broadcasts a request specifying type of data (interest) and the region. As the request flows out into the network, a gradient is set up to flow data towards the collector. Each node has an interest cache. Each interest entry contains immediately previous hope for that interest, gradient direction, and the data rate for that gradient. Each node could have a direction and data rate for each of its neighbor. When a neighbor reports detecting data of interest, the data rate associated with that neighbor is increased. This way the protocol reinforces best paths from sources to the collector. While this approach works well for functions such as detecting motion or to track trajectory of an object, it does not work efficiently to periodically collect data from the network. In the initial stage before the proper data rates for the gradients are sent, the collector can receive same piece of data from multiple neighbors. The protocol does suggest performing aggregation on route to the collector, but it is designed more for moving pieces of information between regions of network then collecting data from the network.

Another approach to collecting data is the tiny aggregation (TAG) approach developed at UC Berkeley [14]. In this approach the collector sets its level equal to 0. It then broadcasts a route tree creation message that contains its id and level. Each node that receives this message sets the sender as its parent and sets its level to be one more than the sender's level. It then rebroadcasts the message with its own id and level. Each node sets the node from which it first heard the request as its parent and ignores all subsequent requests. This propagation will continue until all nodes have received a request. The nodes also listen for rebroadcast of the request by their children. Each node can keep a list of its children. If a node does not hear retransmitting of its request after a certain time period assumes that it is a leaf node. The leaf nodes start the collection by sending their data value to their parents. Each parent node waits for a certain amount of time for their children to report their value. They combine their own value with their children's values and send the results to their parents. This way the results are propagated to the collector node. The protocol also uses snooping to improve efficiency. When computing max or min if any leaf node or a parent node that has collected from all its children hears another node report a value higher(for max) or lower(for min) than its report value, that node does not report its value to its parent. This protocol is the most efficient in terms of hops of all collection protocols because after the set up of the collection tree each node has to transmit only one message for each collection. One of the drawbacks of this protocol is that it requires each node to keep state information. Each parent node has to know its children and every node including the leaf nodes have to know id of their parents. There is also a latency problem because each parent waits for its children to report. If one of the nodes dies or moves away, the collection data from all nodes below it could be lost and the collection from the nodes above would slow down because of the waiting. TAG approach provides a very energy efficient protocol but has latency and reliability problems.

Concast and TAG use similar approaches to the data collection problem. They both disseminate the collection request through the network using broadcast. Both protocols set up a collection tree and perform data reduction at the parent nodes with multiple children. Each of these nodes that can perform reduction is required to keep state information about the type of data being collected and current status of collection. In the TAG protocol, the tree is set up when the request is being sent through the network, while in Concast tree is set up as the data is sent back to the collector and Concast nodes learns about nodes down the tree. The main difference between the two protocols is that in TAG parent nodes waits for all of its children nodes to send their data and then perform reduction on the data. In Concast, parent nodes perform reduction based on previously forwarded data and does not wait for all child nodes to reply before forwarding data. This makes TAG more efficient in terms of number of transmissions while Concast could perform collections in less time for some networks.

Dynamic source routing (DSR) protocol described in [2] gives an efficient method to discover routes and send messages between two nodes in a wireless network. Routes are discovered by sending out a route request broadcast until it reaches the destination node or a node that knows the route to the destination node. This discovered route is included in each message sent between the two nodes. The intermediate nodes simply find themselves in the route and forward the packet to next hop. This protocol does not require nodes to keep routing tables as in the distance vector routing protocol described in [3] and is thus suitable for resource limited sensor nodes. Although this protocol has been shown to work well in one to one transmissions in wireless network, it is not by itself very efficient for many to one flows. Discovering routes to all the nodes and then receiving individual messages from each node would be too expensive and cause too many collisions. There have been many papers on efficient transmissions between nodes in a wireless sensor network, but there has not been a satisfactory solution to the problem of collecting some aggregate data from all or part of a large wireless sensor network.

CHAPTER III

PROBLEM STATEMENT

The two main aspects of this thesis are as follows:

(1) Investigate and implement various ways to randomly place sensor nodes in a given region.

(2) Design, implement, and evaluate a protocol for efficient data reduction in large wireless sensor networks.

These implementations will be done on top of the wireless network simulator designed by Dr. Riley. The random node placements are done to evaluate the performance of the protocol for different types of sensor networks. The placement has to be random enough to get a realistic test of the protocol, but it should also be consistent to allow comparison of performance of different protocols on random networks. There are two approaches to node placement: non-clustered and clustered. In non-clustered mode, the same random distribution scheme is used throughout the region. In clustered mode, centers for clusters are chosen randomly in the region and a point that is closer to one or multiple cluster centers has a higher probability of having a node. [4] describes two different cluster models:

(1) Quadrat-based model: The whole area is partitioned into non-overlapping regions. The center of each region has a certain probability of being chosen as cluster center. The size of the cluster is equal to the size of the region.

(2) Center-satellite model: Cluster centers are chosen such that each point in the area has a certain probability of being chosen as a center based on some random distribution. Each center can have either a fixed size region of influence or sizes of regions can be determined randomly. This model allows for overlapping regions.

This thesis implements some common non-clustered and clustered random distributions for the nodes. Various placements schemes are used to evaluate the performance of the protocol for different types of networks.

The design and implementation of the protocol for data reduction is done on top of DSR, which was implemented in the simulator by another set of students. The main considerations for the design were scalability, minimizing amount of data transmissions, minimizing latency and maximizing performance. The goal was to provide better performance for data reduction than any current protocols. The performance is based on amount of time taken to collect data and the number of packets and number of hops required to perform a collection. This assumes that the energy spent to transmit data over a single hop is about the same throughout the network. The performance of this protocol is compared with performance of a base protocol where all the sensor nodes send their data directly to the collection node when a request is broadcasted. The performance of the protocol is measured for different random distributions of nodes and different size of networks. The results from different node placement schemes are compared to see if the protocol works better for some particular types of networks. The protocol is also designed to minimize the amount of state stored by each node and to minimize the amount of processing done at each node. This is done because of limited memory and processing resources available at the sensor nodes. The protocol uses a hierarchical approach with different algorithms for lowest level and upper levels. A three-level network is implemented and tested for performance, which is then generalized for more levels. The protocol is evaluated based on the amount of load it induces on the network as compared to the naive approach of each sensor node sending its data directly to the collector. The protocol is also compared with Concast protocol described in [7] and the tiny aggregation protocol described in [14]. Only minimum required functionalities of these two protocols are implemented to accurately compare the protocols.

CHAPTER IV

DESCRIPTION OF DATA REDUCTION PROTOCOL

The problem of data collection is solved by using a multilevel hierarchy and data reduction when possible. Two different protocols are used, one for the lowest level in the hierarchy and one for upper levels. These protocols rely on existence of DSR protocol in the simulator.

4.1 Lowest Level Protocol

The network is divided into predefined geographical regions. Each node knows the number of its region when it is placed. The lowest level protocol describes the method to collect from all nodes in a single such geographical region. The protocol assumes that a route between any two nodes is or can be generated using DSR, and the network is connected. The network is assumed to be symmetric meaning if a can send message to b then b can send message to a. Using the routes in its DSR cache, the collector node generates spanning paths that go through every node in the region at least once. The collection requests are sent along these paths and replies are sent back along the reverse routes. Data reduction is done on-route in the reply messages. There are three types of nodes in the protocol: one collector node, leaf nodes of the spanning paths, and the intermediate nodes. The following is the specification of processing done at each type of node and the type of messages sent between these nodes:

4.1.1 The Collector Node

(1) If not already known, find the ids of all the nodes in the region. The process to do this is described later.

(2) Generate spanning collection routes:

(a) From the route cache, take the route with highest number of nodes that are in the collector's local region as the first path. Also construct and store a bit map that is the same size as the length of the route. The bitmap has a 1 corresponding to each node covered by this path.

(b) While at least one node is not covered by a path:

> Add a path that has the highest number of nodes not already covered by one of the paths
>
> If there are not any unselected paths in the route cache that have an uncovered node, initiate DSR route discovery to an uncovered node and then add the discovered path

(c) The collection paths and bit maps indicating which nodes are covered by each path could be stored by the collector or reconstructed each time based on the paths in the route cache

(3) Send out collection request messages along the spanning paths constructed in the previous step. The request message contains a request id, level, a sensor id indicating the type of sensor data, and an opid indicating the type of reduction operation to be done over the data. The collector also stores these three values along with a current collection value. The current collection value is initialized to the

sensor measurement from the collector. The collector also stores a reply counter initialized to the number of spanning paths.

(4) Process the reply messages: The collector uses the request id in the reply message to access data corresponding to that collection. The data in reply message is combined with the current collection data according to the specified reduction operation. The reply counter is decremented by one.

(5) When the reply counter reaches 0, collection has been done from all the nodes. The final value is reported and the state data corresponding to this request is removed.

4.1.2 The Leaf Nodes

(1) Upon receiving a collection request message, initiate a reply message. The reply message consists of the request id, sensor id, and opid from the request message. The reply also consists of the sensor data from the leaf node. The bit-map is also sent along with the reply message after clearing the bit corresponding to the leaf node.

(2) The reply message is sent back to the collector by reversing the path of the collection request.

4.1.3 The Intermediate Nodes

(1) Forward the request messages to the next hop in the source path. One optimization could be to keep the sensors off for most of the time. The intermediate nodes can turn on the appropriate sensor upon receiving the request message if there is a one corresponding to this node in the bitmap.

(2) For the reply message:

If the bit corresponding to this node is zero

```
        Forward the packet to the next node in the route
    Else
        (a) Combine this node's data with the data in the packet as specified by the
            opid and set bit corresponding to this node to 0
        (b) Forward the packet to the next node in the path
```

To ensure that all the nodes in the region are covered by spanning paths, the collector has to find out the ids of nodes in its region. This could be done by using expanding ring broadcasts as described below:

(1) Broadcast a region discovery message containing the requester's id and region number. The ttl of this message is set to 1.

(2) All nodes receiving the discovery message check their recent discovery cache. If the requester's id is not in the cache and requester is in receiving node's region, nodes send a reply to the requester with its id. Nodes also save the id of the requester in recent discovery request cache.

(3) Upon receiving a reply, the requester checks the list of ids. If the id is not in the list, it is added to the list. After a certain time T, requester reissues the request message after incrementing the ttl. This is done until no new nodes are found in the last n requests. The values of T and n are determined the user and should be based on the type of network and amount of reliability desired.

Data reduction schemes are used by the protocol if the type of data allows such reduction (min, max, avg, etc.). To collect data such as median that do not allow reduction, each node places its data in place of its source address in the source route. If the data being collected is larger than the size of the source addresses, it can be placed in the data portion of the packet in the same order as the order of nodes in the route.

An alternative implementation is to store the spanning paths and bitmaps at the leaf nodes. The collection request message can then be multicast to the leaf nodes. The leaf nodes can then send the reply message along the spanning paths. The multicast of collection request should be implemented using the most efficient existing multicast protocol. This could be an optimization for a relatively static network.

4.2 Higher Level Protocol

Multiple lower level regions are grouped together to form a higher-level region. For example for a K greater than 1, a K level region would consist of a group of K-1 level regions. Each node is informed about the number of lower level regions at each level. A collector is chosen at the highest level of hierarchy from which collection is to be done. In this description, this highest level will be referred to as level n. The n level collector finds and constructs spanning paths through n-1 level collectors. The n-1 level collectors perform collection in their n-1 level regions and then send results back to the n level collector through reverse spanning paths. The processing done by each type of collector and messages exchanged between them are described below:

4.2.1 The n-level Collector

(1) The collector generates spanning paths through all of the n-1 level regions. Spanning paths are generated such that at least one node from each of the n-1 level region falls in one of the routes. This could be done in two ways:

A. Discovery of a single Shortest Route through all the lower level regions:

a. Mark the n-1 level region of n-level collector node as covered

b. Generate a path to a node in an uncovered n-1 level region that is closest to the last added collection node

c. Add this path to the route and mark that closest node as collection node for its n-1 level region

d. If all n-1 level regions are not covered, go back to step b

B. Discovery of multiple routes that span at least one node in each region

a. Mark the n-1 level region of this node as covered

b. From the route cache, iteratively add paths that go through a node in highest number of uncovered n-1 level regions, if all n-1 level regions are not covered go to step c

c. Generate Paths to closest nodes in one of the uncovered n-1 level regions using DSR. The closeness would be determined using the number of hops needed to get to a node. The details of this are discussed below.

d. Add the path that goes through the highest number of uncovered n-1 level regions to the group of collection paths

e. Select one node from each uncovered n-1 level region in the path as the collection node for that region

f. If all the n-1 level regions are not covered, go back to step c

The first algorithm would only work well when the size of the network is small and regions are placed such that each region has at least one node only a few hops from the collection node. This method does not scale well. Since scalability is one of the criteria for this simulator, the second method is used in the protocol. Following is the method to accomplish Step c of method B:

a. The collection node sends out a broadcast with TTL equal to one hop and a list of n-1 level regions already covered.

b. All nodes receiving the request whose n-1 level region number is not in the data packet send a reply back to collection node containing its address and its n-1 level region number.

c. The collection node stores the address along with the n-1 level region number in its cache. If multiple messages are received from a region, the one that is the least number of hops away is chosen.

d. If all n-1 level regions are not covered, go back to step 1 after incrementing TTL by one and placing numbers of all contacted n-1 level regions in the data packet.

The n-level collector also acts as the n-1 level collector for its n-1 level region. It executes the lowest level protocol if n-1 is the lowest level; otherwise, it executes this same code for n-1 level.

(2) The n-level collector sends out collection requests over the spanning paths. The collection requests contain request id, sensor id, opid, and level of collection being done. The requests contain a timeout after which the request is dead. The requests also contain a bitmap with 1 indicating the lower level collector nodes. The collector also stores the id values along with the current collection value that is initialized to the measurement

from this collector. A reply counter initialized to number of spanning paths is also stored.

(3) Process the reply message: collector uses the request id in the reply message to access data corresponding to that collection. The data in reply message is combined with the current collection data according to the specified reduction operation. The reply counter is decremented by one.

(4) Check for unfinished collectors: For the n-1 level regions that were unable to give their data, collection node can reissue the request using the returned bit map as the bit map for the request. Another option is for the collection nodes of unfinished regions to store higher-level collection node's address, and when data becomes available, send it directly to that collection node. For this second option, the higher-level collection node would increment the reply counter by the number one's in the bitmap from the reply message.

(5) When the reply counter reaches 0, collection has been done from all the nodes. The final value is reported and the state data corresponding to this request is removed.

4.2.2 Intermediate Collector Nodes

(1) Process the Request Message:

```
If there is a zero in the bit corresponding to this node in the request message
        Forward the request message to the next hop in the route
  Else
                    i.  Get the request level from the message
```

ii. If this level is greater than 1 then execute the n-level collector protocol described above with n = level –1. Otherwise execute the code for collector in the lowest level protocol.

iii. Forward the message to the next hop in the route

(2) Process the reply message:

If there is a zero in the bit corresponding to this node

Forward the reply message to the next node in the path

Else if data collection from its region is done

(a) Combine the region's data with the data in the message as specified by op id and set its bit to 0

(b) forward the message to next node in the path

If data collection is not done in its region:

(a) Store the reply message for some amount of time, e.g.

Tstore = (1/f)* Timeout, where Timeout was specified in the original request, and f is specified by user

(b) When the data becomes available, forward the packet to next node after placing its data and setting its bit to 0

(c) If data does not become available before time is over, forward the message to next node without setting its bit to 0 and store the address of the n-level collector.

(d) If the higher level collector reissues the request for unfinished n-1 level regions, store the data and wait for the request. Otherwise, send a unicast message containing the collected data to the higher level collector node.

4.2.3 Leaf Collector Nodes

(1) Process the Collection Request:

a. Get the request level from the message

b. If this level is greater than 1 then execute the n-level collector protocol described above with n = level –1. Otherwise execute the code for collector in the lowest level protocol.

c. Store the reverse source route along with the request id. Associate the request id of the lower level collection with the request id in the request packet.

(2) Initiate the reply message: The reply message contains reverse route and reverse bitmap from the collection request message. This message is initiated after the data has been collected in the lower level region of the leaf node. The leaf collector node puts its collected data in the message, sets its bit to 0 and forwards the message to the next node on the reverse route.

The non-collector nodes in the spanning routes would behave just as they did in the description of the lowest level protocol. Although having nodes temporarily store reply messages when they are not done collecting is not the ideal solution, it should not occur very often due to relatively equal sizes of the regions. The storing might have to be done in the first few nodes in the return path, but the collection nodes in the regions towards the end of the return path would have had more time to collect their data. The option of the unfinished collector node sending a unicast directly to the higher level collector is implemented when evaluating the protocol. Since the unfinished region has a good chance of being towards the end of the collection path, this option seems better, but it requires the node to temporarily remember the collection request and address of the collection node. If one of the nodes on the spanning path goes down, the previous node will try to find a route to the node after the failed node. The collector is then notified of this new path and the set of spanning paths will be modified. This fault and mobility tolerance can be built in as part of DSR.

CHAPTER V

IMPLEMENTATION OF NODE PLACEMENT ALGORITHMS

5.1 Non-Clustered Distributions

Several schemes were considered to randomly distribute the sensor nodes in an entire given region. Four non-cluster distribution schemes were implemented as described below.

5.1.1 Bounded Normal Distribution

In this distribution, the number of nodes, variance, width and height of the region are specified. Random x and y coordinates are generated using normal distribution with the specified variance around the center of region. If the generated coordinates fall outside of the size of the region, they are thrown away and other coordinates are generated. This throwing away does not seem have a significant effect as long as the variance is relatively small compared to the size of the region. This process is iterated until coordinates for all the nodes are obtained. The normal points are generated using the following polar method described in [1]:

(1) Generate U1 and U2 as two uniformly distributed random variables

(2) Set $Vi = 2Ui - 1$; and $W = V1^2 + V2^2$. If $W > 1$ go back to step 1

(3) $Y = ((-2 \ln W)/W)^{1/2}$

(4) $X1 = V1*Y$ and $X2 = V2*Y$ are two normally distributed random variables.

A sample of a randomly generated region is given below:

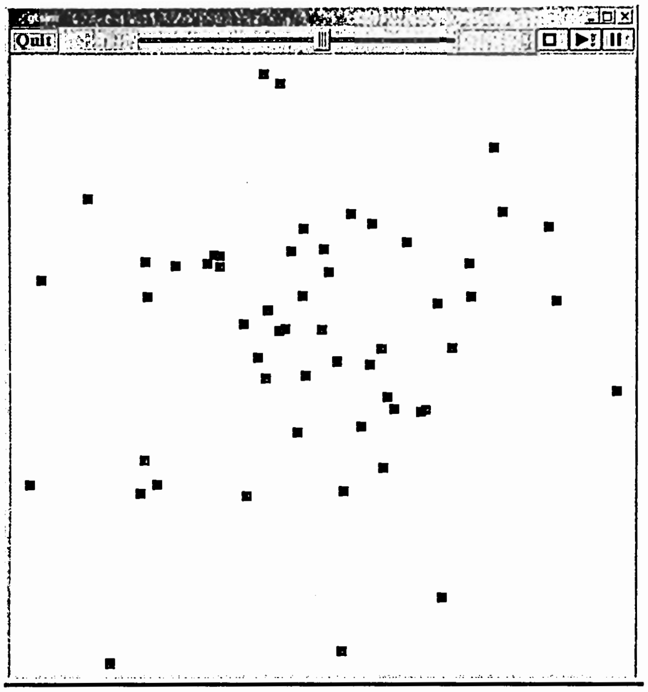

Figure 1: Bounded Normal Distribution

5.1.2 Uniform Distribution

In this distribution, the number of nodes, height, and width of the region are specified. Two uniformly distributed coordinates within specified ranges are generated and used as the location of the nodes. This distribution was implemented by Dr. Riley. A sample of region created by this distribution is given below:

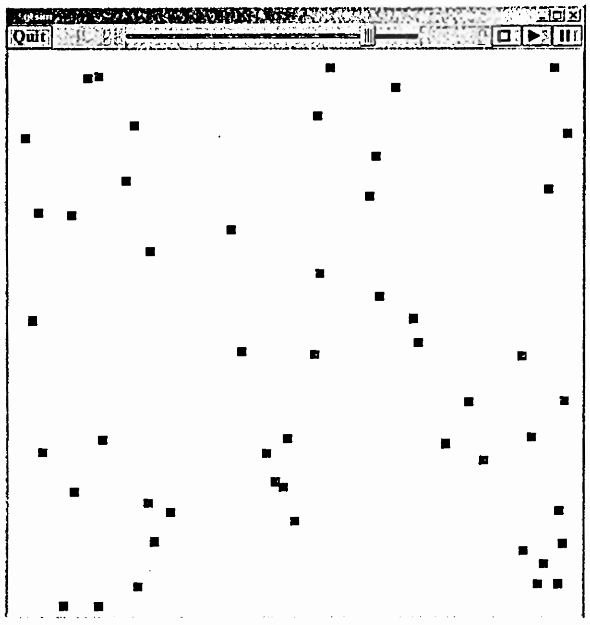

Figure 2: Uniform Distribution

5.1.3 Radial Distribution

In this distribution nodes are distributed at a random angle and random radius from the center of the region. This distribution was implemented by Dr. Riley. The angle and radius can be generated using different distributions. First a random radius r and a random angle A are generated. Then the location of the node is computed by using the following formula:

X = r * cos((A * 2.0 * PI)/360.0)

Y = r * sin((A * 2.0 * PI)/360.0)

When the angle is distributed from 0 to 360, the nodes are distributed in a region defined by circle of radius R, where R is the bound on the radius. A sample of region with this distribution is given below. In this sample, the radius was distributed uniformly from 0 to R and the angle was distributed uniformly from 0 to 360.

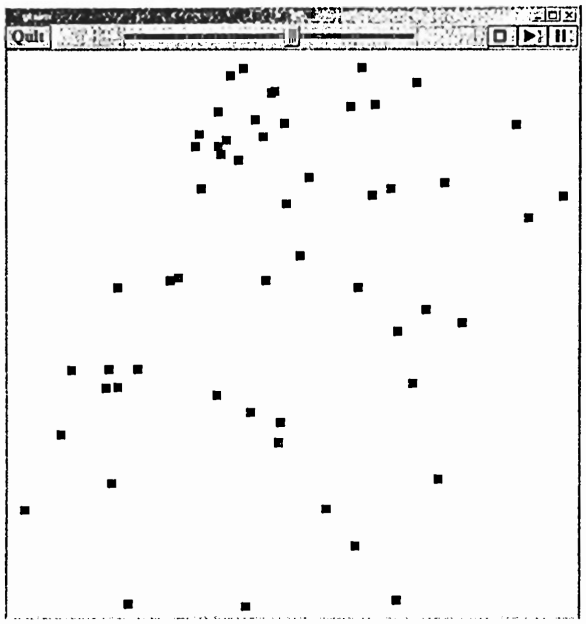

Figure 3: Radial Distribution

5.1.4 Poisson Distribution

In this distribution, node density of the region is specified instead of specifying the number of nodes. The region is divided into sub-regions of the same area. The area is determined such that the probability of having two nodes in that area given the density is less than 0.001. If this area comes out to be less than one, an area of size one is used (each region is a single point). Using the density of the region and area of sub-regions, the probability of having a node in a sub-region is computed. For each smaller region, a uniform random number between 0 and 1.0 is generated. If this number was less than the probability for the region, a node was placed at the center of the region. This distribution would look similar to the uniform distribution. A sample region is shown below:

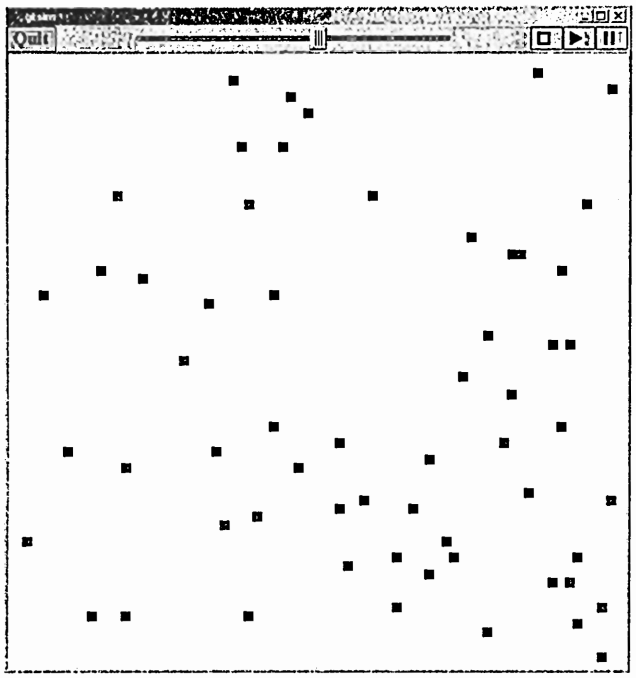

Figure 4: Poisson Distribution

5.2 Clustered Distributions

Instead of placing nodes using the same random distributions throughout the region, nodes can be placed into clusters using some random distribution. Centers are placed using some random distribution around the region. Each center will have a specified square region of influence given by its width and height, which can be the same for all centers or determined randomly. These influence regions could be overlapping. Sensor nodes would be placed in these influence regions using a specified random distribution. Following Steps would be followed to generate Clusters:

- Step 1: Pick cluster centers according to uniform, poisson, radial, or bounded normal distributions described above. The inputs for these functions would be the overall region width, height, and the origin as the center.

- Step 2: For each cluster center call the uniform, poisson, radial or bounded normal functions described above, specifying width, height and center of the cluster to generate coordinates for the nodes.
- Step 3: Examine the nodes created in Step 2 and throw away all the nodes that fall outside of the whole region. This would not be a problem for uniform, radial and poisson distributions; however for normal distribution this throwing away of nodes could give strange looking distribution. To compensate for this, a lower value of sigma would have to be used depending on the size of the cluster that fits inside the big region. This would implement the center satellite model.

For Poisson distribution, each cluster could be created with same density, or the density of each cluster could be chosen using some random distribution. Since there are four distribution choices for the cluster centers and four choices for distribution in the clusters, sixteen different types of clustered distribution can be generated. Since the uniform distribution, radial distribution and the Poisson distribution give similar types of regions, only two of the clustered distributions were implemented using uniform and bounded normal distributions. The first clustered distribution was generated by distributing the centers uniformly and then uniformly distributing nodes around those centers. Following is a sample of a network generated by this clustered distribution:

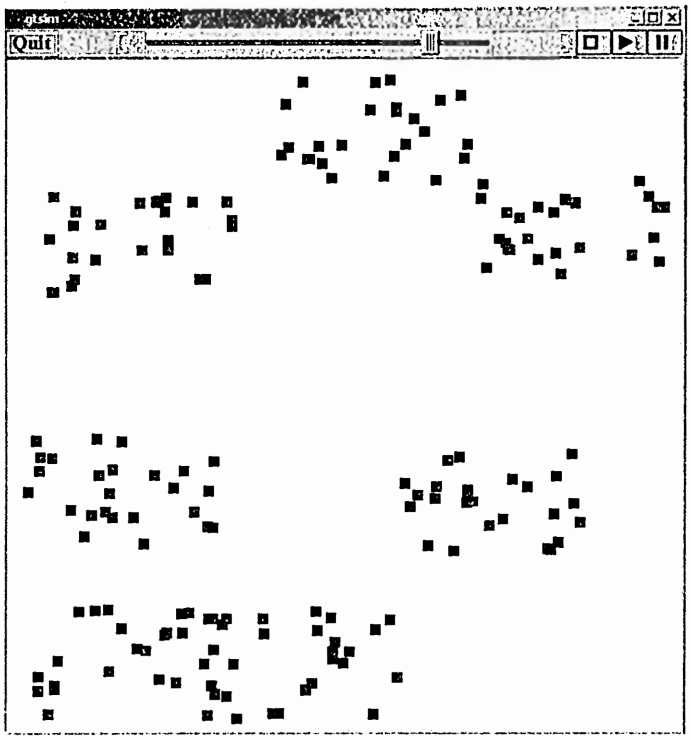

Figure 5: Uniform-Uniform cluster distribution

This sample network has seven total clusters. The two clusters at the bottom overlap with each other. The second clustered distribution was generated by distributing the centers uniformly and then distributing the nodes around this centers using bounded normal distribution. The following is a sample of a network generated by using this clustered distribution. There are eight total clusters, but two of cluster overlap and give the big cluster on top left.

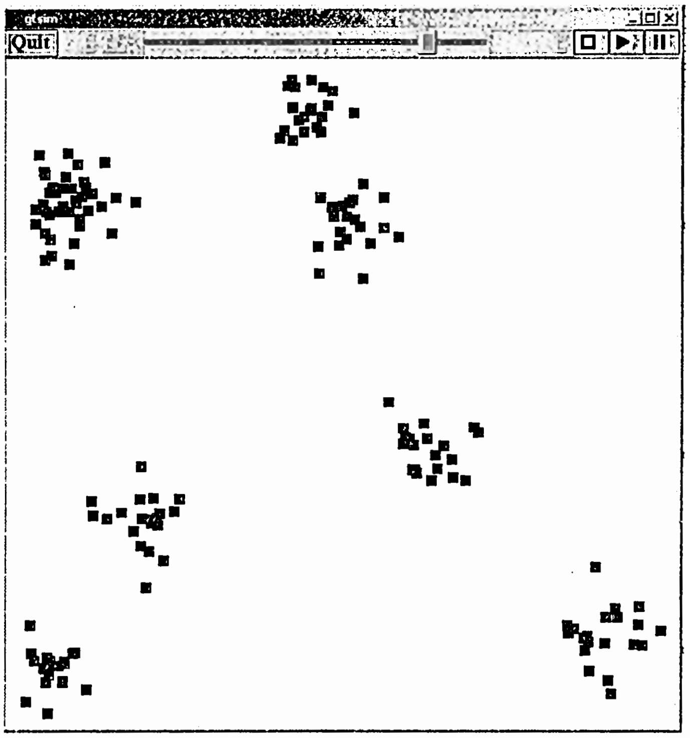

Figure 6: Uniform-BoundedNormal Distribution

CHAPTER VI

IMPLEMENTATION OF DATA COLLECTION PROTOCOLS

6.1 Data Reduction Protocol

The data reduction protocol described in chapter 4 was implemented over the GTNet simulator designed by Dr. Riley at Georgia Tech. The implementation follows closely to the protocol description. Multiple spanning paths were used for higher level collection. If a collector does not have path to any nodes in an uncovered region, a route discovery was done to a default node in that region. This was done to simplify the implementation. In actual implementation an expanding ring search would be done to find a node from that region as described by the protocol. If a region is not done collecting by the time the return packet gets to the collector, the collector forwards the packet and sends a direct packet to higher level collector when the collection is done. This never occurred during the simulations. Lower level collectors were always done collecting by the time the reply message got back to them. Each node was assigned a location upon placement using one of the random distribution schemes. Sixteen equal size lowest level regions were based on fixed geographical boundaries. Each node can determine its region by knowing the size of the regions and its own location. Four lower level regions were grouped together to form a higher level region, so the network consisted of sixteen 1-level regions, four 2nd level regions, and one 3rd level region. A picture of a sample network is given below:

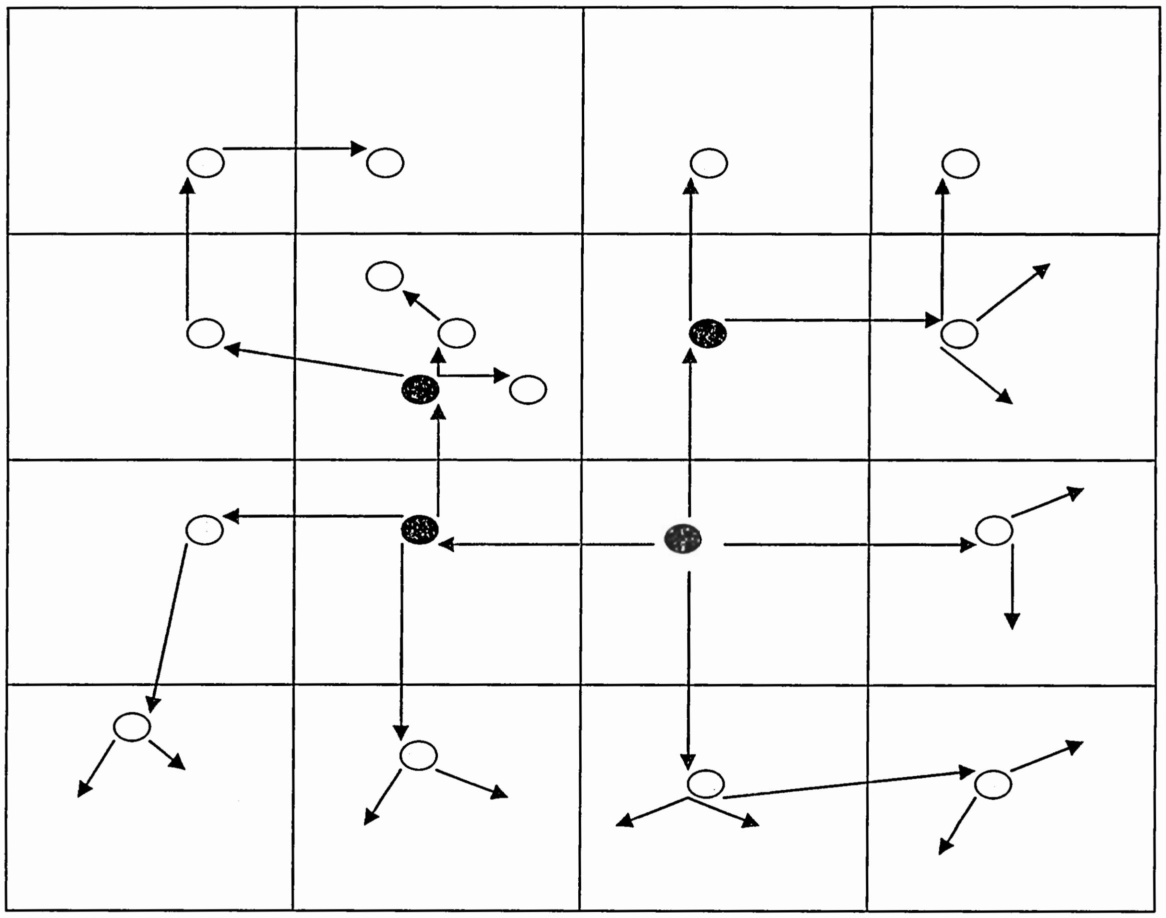

Figure 7: Data Reduction Protocol over a Sample Network

In this sample network, the orange node is the 3rd level collector and the orange paths are spanning paths for the third level. The blue nodes are the 2nd level collectors and the blue paths are 2nd level spanning paths. The yellow nodes are 1st level collectors and black arrows indicate local level spanning paths. The uncolored nodes are non-collector nodes and are only shown for one of the regions. The protocol is called as a hook function from the TCP/IP layer. This is done because the protocol has to be executed at some intermediate nodes in the tcp transmission as well as the end nodes. TCP packets are used to send the collection requests and responses. Each collection request contains the type of collection and an id. The id and the collector's address differentiate different collections. Only the collector nodes have to keep track of the higher level collector address and the id. The non-collector nodes are not required to keep any state information. The response packet from lower level collectors contains the number of packets and hops taken to perform the collection in that region. This data is used for evaluation purposes.

6.2 TAG and Concast Protocols

The minimum functionalities of these protocols were implemented. The collector sends a broadcast to set up the collection tree. Each node receiving the broadcast for the first time sets the source as its parent and rebroadcasts the message. Each node then sends a packet with its data to its parent. For Concast, the parent nodes forward the data as soon as they receive them. When the aggregate being collected is min or max, the Concast parent nodes keep track of lowest or highest value forwarded and ignores any value that is higher or lower than that forwarded value. For tiny aggregation, each parent is notified of the number of children it has. The parent node waits for a certain amount of time to get the data from all of its children, and then forwards the combined data to its parent. The amount of wait increases higher up on the tree. The protocols are integrated into the IP layer. UDP packets are used to send tree-setup broadcast and the data packets.

CHAPTER VII

ANALYSIS OF THE RESULTS OF THE DATA COLLECTION ALGORITHMS

One of the main goals of the thesis was to design a new data collection protocol and then compare the performance of the protocol against existing collection protocols. The main parameters for performance are amount of energy required per collection measured by number of hops and packets, and the amount of time required to perform a collection. The amount of state required by each protocol is examined by evaluating the requirements of the protocol. This chapter compares the performance of the three protocols mentioned in Chapter 6. In addition a general naïve approach was also implemented and tested. In this approach, the collector broadcasts the request through the network, and each node receiving the request sends a unicast reply back to the collector. This protocol does not do any processing at intermediate nodes. Tests were run on networks with uniformly distributed nodes and then on networks with bounded normally distributed nodes. The performance was measured on Dr. Riley's GTNetS simulator running on enzo cluster. The processor node for the simulation consisted of two 847.258 MHz Intel Pentium III processors. Amount of memory available was 2.06 GB. The simulations were run on Linux operating system version 2.4.18.

7.1 Performance Tests over Uniformly Distributed Networks

In each of these tests, nodes were distributed uniformly over a 400 by 400 region. For the data reduction protocol, the network was divided in sixteen 100 by 100 geographical regions. The data reduction protocol was evaluated using three levels with four lower level regions per each higher level region. Four lowest level regions were combined to give one second level region. The four second level regions were combined to give a single third level region. The set up was as described by Figure 7. The radio ranges (transmission range) of the nodes were set to 40. In the initialization stage, n/2 messages were exchanged between random source/destination pairs, where n is the number of nodes in the network. The amount of time, packets, and hops required to perform a single collection through the network were measured. The time included the sending of the request to all the nodes and sending of all the replies to the collector. The time, packets, and hops required to set up the collection trees for Concast and TAG were not counted. The time, packets, and hops required for the third level collector to generate spanning paths were not counted. The times to generate spanning paths at the second and first levels were counted, because these paths are generated on-demand after receiving the collection request. All data was collected by performing a third level collection over the entire network. There are two types of aggregates. Aggregates such as sum, average, count where data from all the nodes have to be reported and aggregates like min/max where accurate results could be obtained without data from some of the nodes. Sum was used to test the performance of the protocol over the first type of aggregates, and min/max were used to test over the second type of aggregates.

7.1.1 Number of Packets Required Per Collection

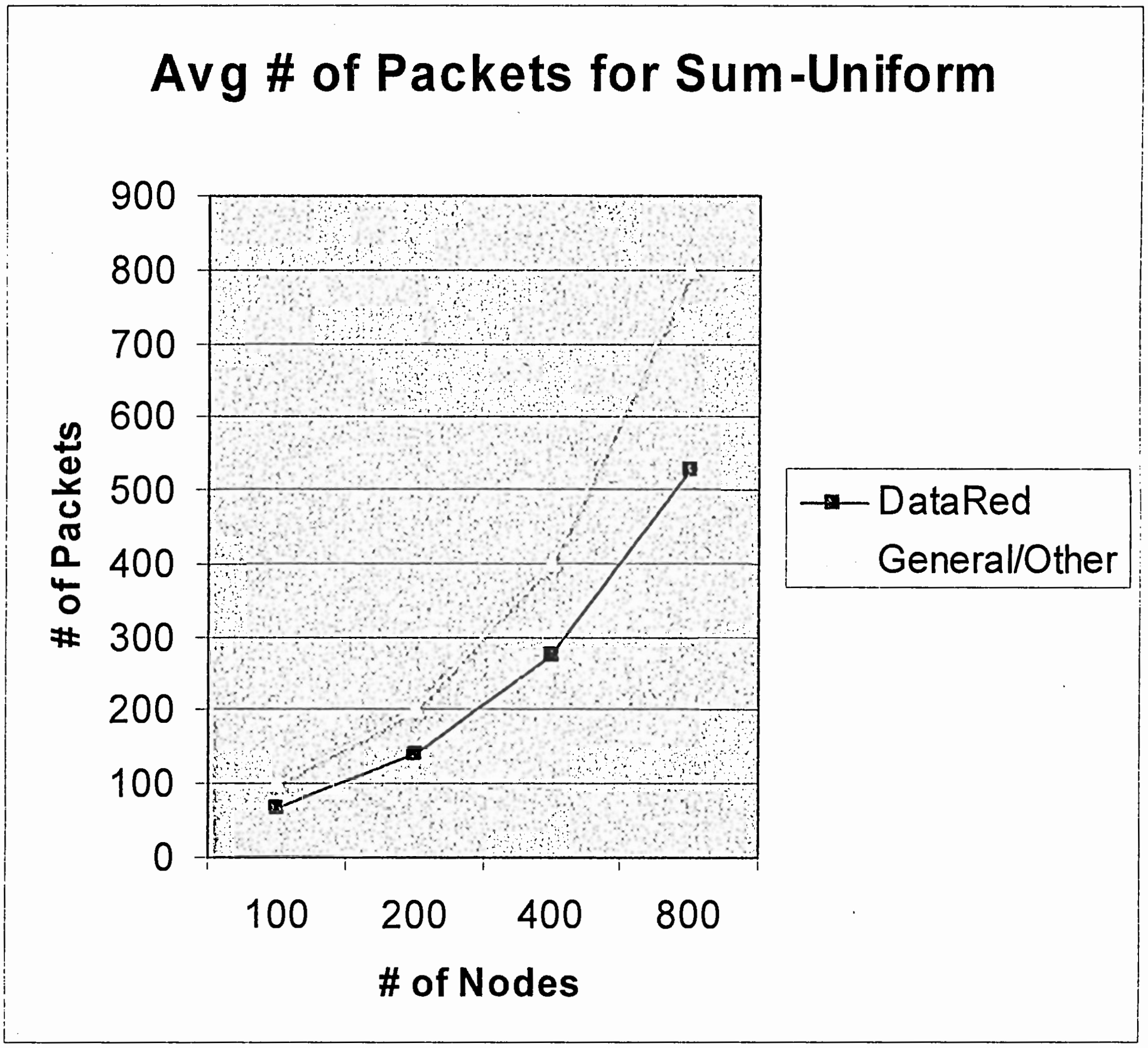

Figure 8: Average Number of Packets per Collection with Uniform Node Distribution

Figure 8 shows the average number of packets required for a single collection for different number of nodes using Data Reduction protocol and the other three protocols. The data was collected by computinga sum over random networks. One hundred random networks were used to test networks with 100 and 200 nodes. Fifty random networks were used to test the networks with 400 and 800 nodes. These same numbers of random networks were used for all the simulation results presented in this thesis. The random networks were created as described above in Section 7.1. For the general approach, Concast and TAG, the number of packets required was equal to the number of nodes, because each node creates and transmits a single packet with its own data. In the data reduction protocol, only the leaf nodes create a packet. This leads to a lower number of packets being created and transmitted as seen in the above graph.

7.1.2 Number of Hops Required Per Collection

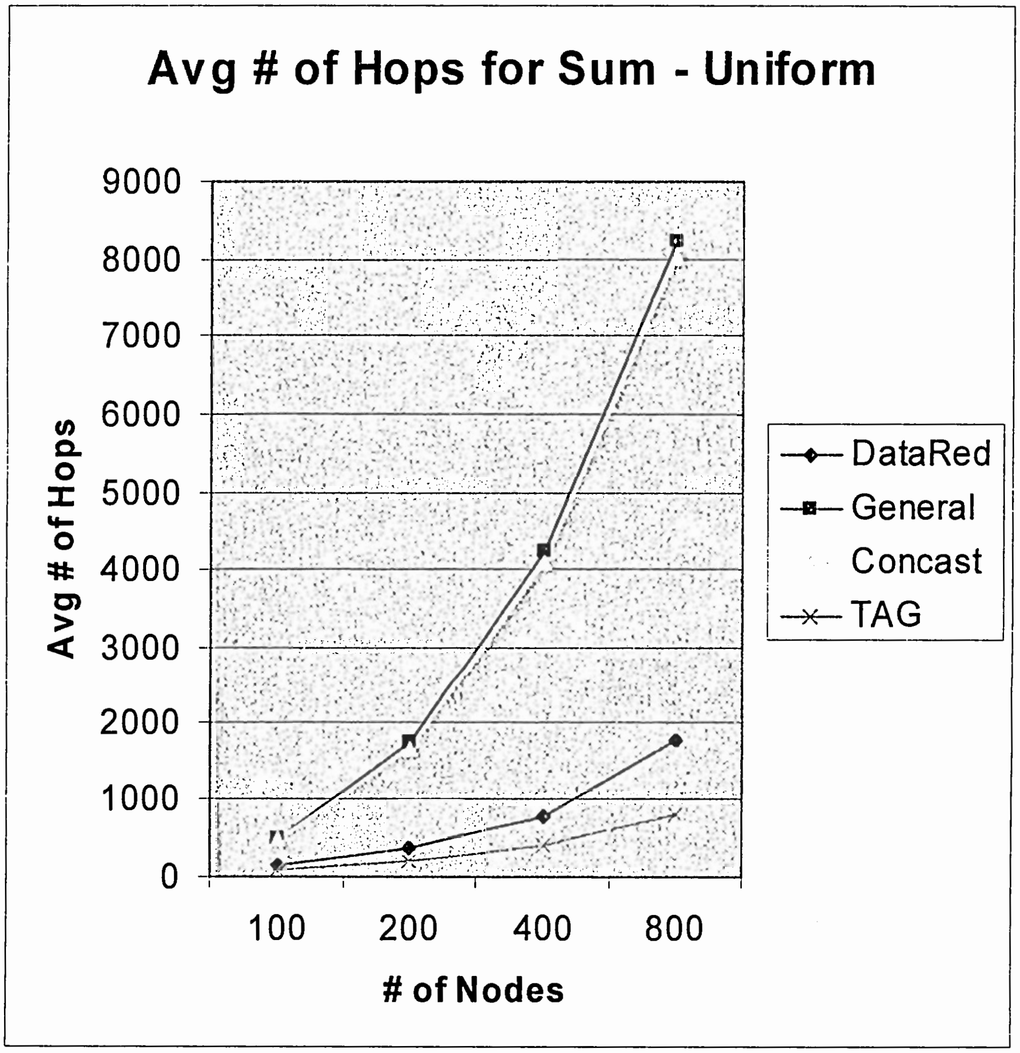

Figure 9: Average Number of Hops per Collection with Uniform Node Distribution

Figure 9 shows the average number of hops required for a single collection for different number of nodes using Data Reduction protocol and the other three protocols. Again the data was collected by computing a sum. As shown by the graph, the data reduction protocol gives a much better performance when compared to the general and Concast approaches. The advantage gets larger as the number of nodes increases. The number of hops required is almost four times lower for the data reduction protocol. In the general approach, the nodes have to pass on data packets from other nodes as well as send their own data packet to the collector. In the data reduction protocol, nodes combine their data into the packets sent to the collector by other nodes. Some nodes still have to pass on multiple packets because they are in multiple spanning routes. In the TAG protocol, the parents wait for all of their children to send their data before sending the aggregate to their parent. For this reason, the number of hops required in the TAG protocol is equal to the number of nodes. Since the collection was done through all the nodes, each node had to transmit at least one time and at least for one hop. Therefore in terms of number of hops, the TAG represents the best possible performance.

7.1.3 Amount of Time Required Per Collection

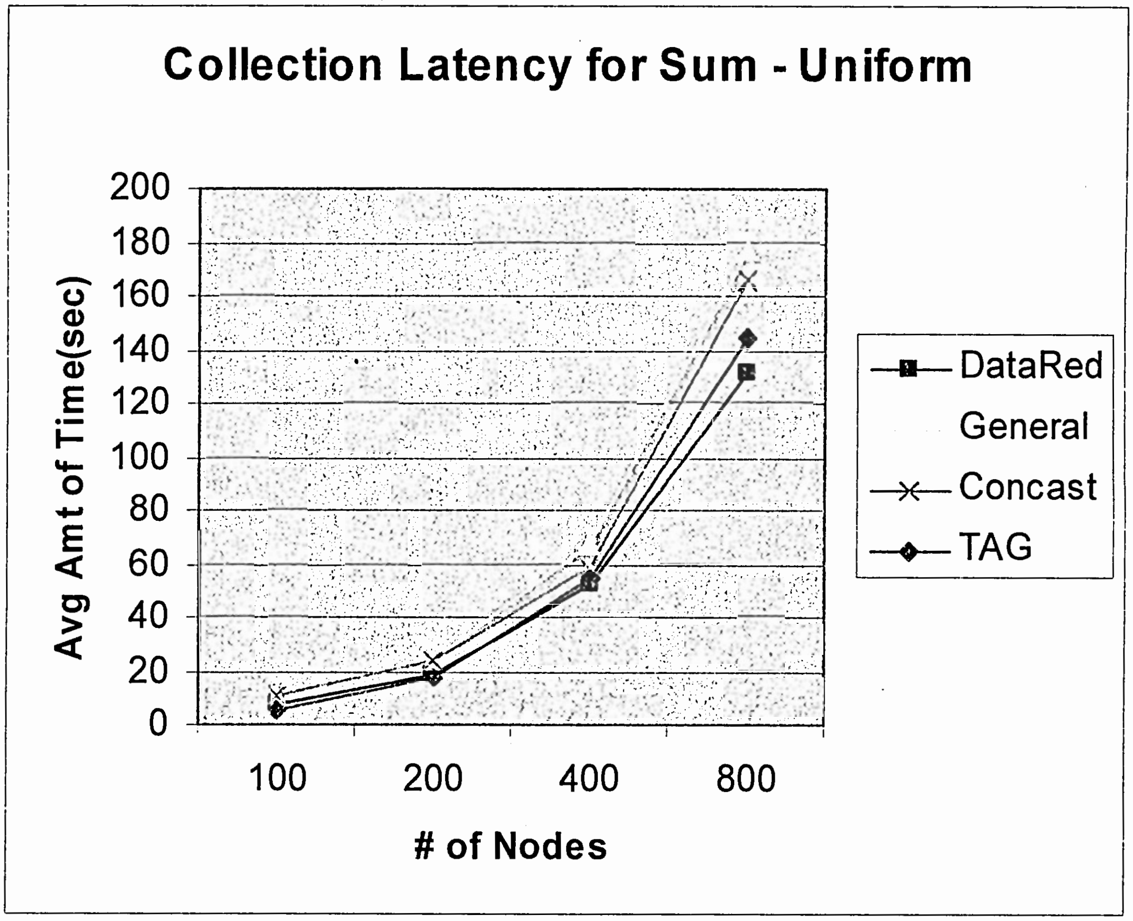

Figure 10: Average Amount of Time per Collection with Uniform Node Distribution

Figure 10 shows the average amount time in seconds per single collection through the network. The time was measured from sending of the request by the final collector to the time when all the data is collected from the entire network. As seen in the graph, data reduction protocol performs better than the Concast and general protocols. This advantage in the amount of time taken is lower than advantage in number of hops and packets. This is because of the time taken to generate the spanning paths and also because the request has to be sent along the spanning paths instead of just being broadcasted through the network. TAG protocol gives a similar performance as the data reduction protocol. For 800 nodes, data reduction protocol performs better than TAG. This is because at high number of nodes there are more children per each parent. Since each parent has to wait for all the child nodes to report, parent node can only report after the slowest child has reported. A delay at a leaf node is propagated to every node up along the tree. All children trying to report at the same time can also cause collisions at the parent nodes.

7.1.4 Number of Packets to Compute Min/Max

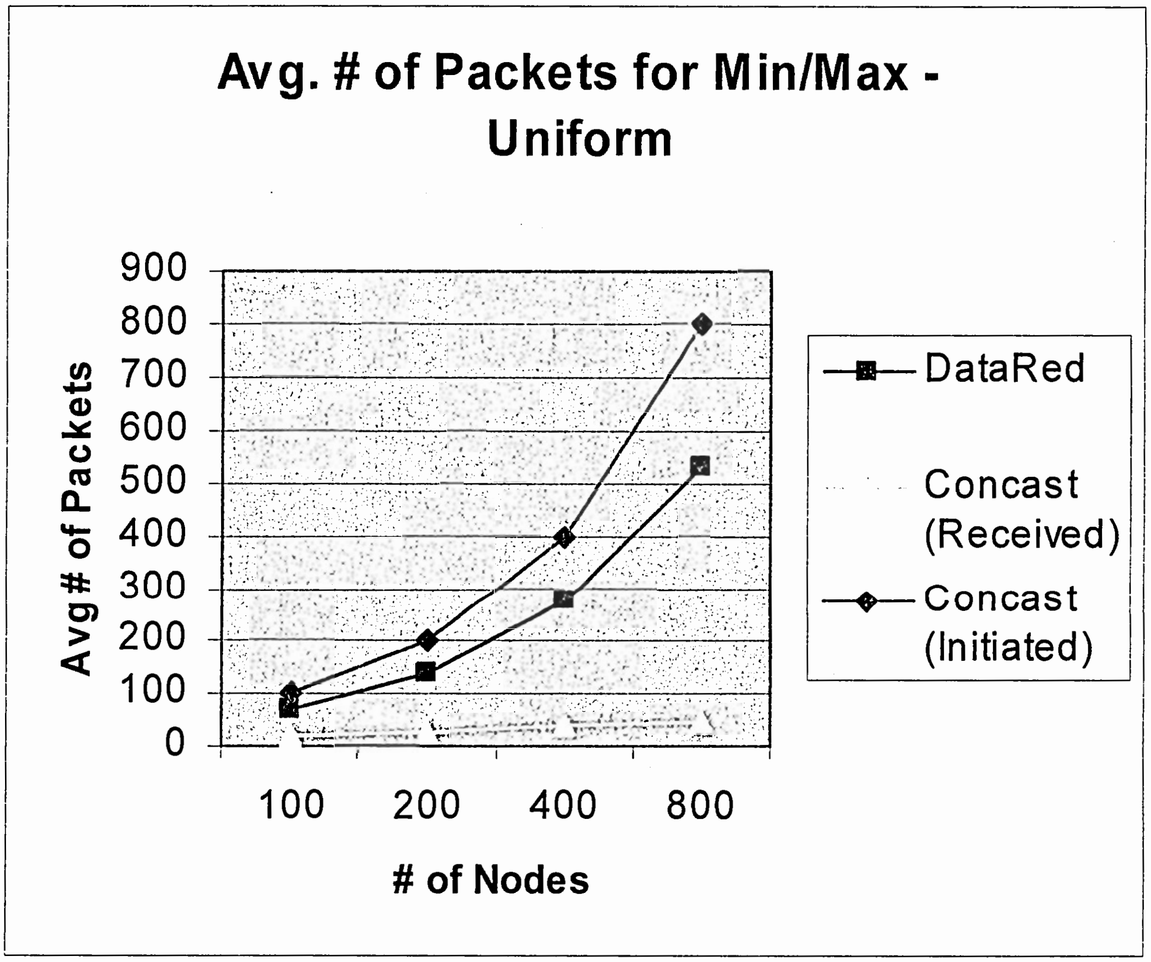

Figure 11: Avg Number of Packets to Compute Min/Max with Uniform Node Dist.

Figure 11 shows the average number of packets to perform min/max computation over a network using Concast and data reduction protocol. The parent nodes in the Concast protocol keep track of the last largest or the smallest value forwarded previously for max or min computation. If the node receives a packet that contains a value lower than (for max computation) or larger than (for min computation) previously sent value, that packet is dropped and not forwarded. The Min and Max operations were tested separately, so the nodes only stored the highest or the lowest value depending on the type of operation. The averages from the two operations were combined to give these graphs. The number of packets initiated is still equal to the number of nodes, but the number of packets getting all the way to the collector is lowered. Each node's sensor measurement value was chosen from 0 to 800 using uniform random distribution. The performance of TAG and the general approach were not measured because both of these protocols process the min/max and sum aggregation the same way. For the TAG protocol, the number of packets received by the collector would be the number of nodes one hop away from the collector. This is because each of these parent nodes would forward only one packet to the collector.

7.1.5 Number of Hops to Compute Min/Max

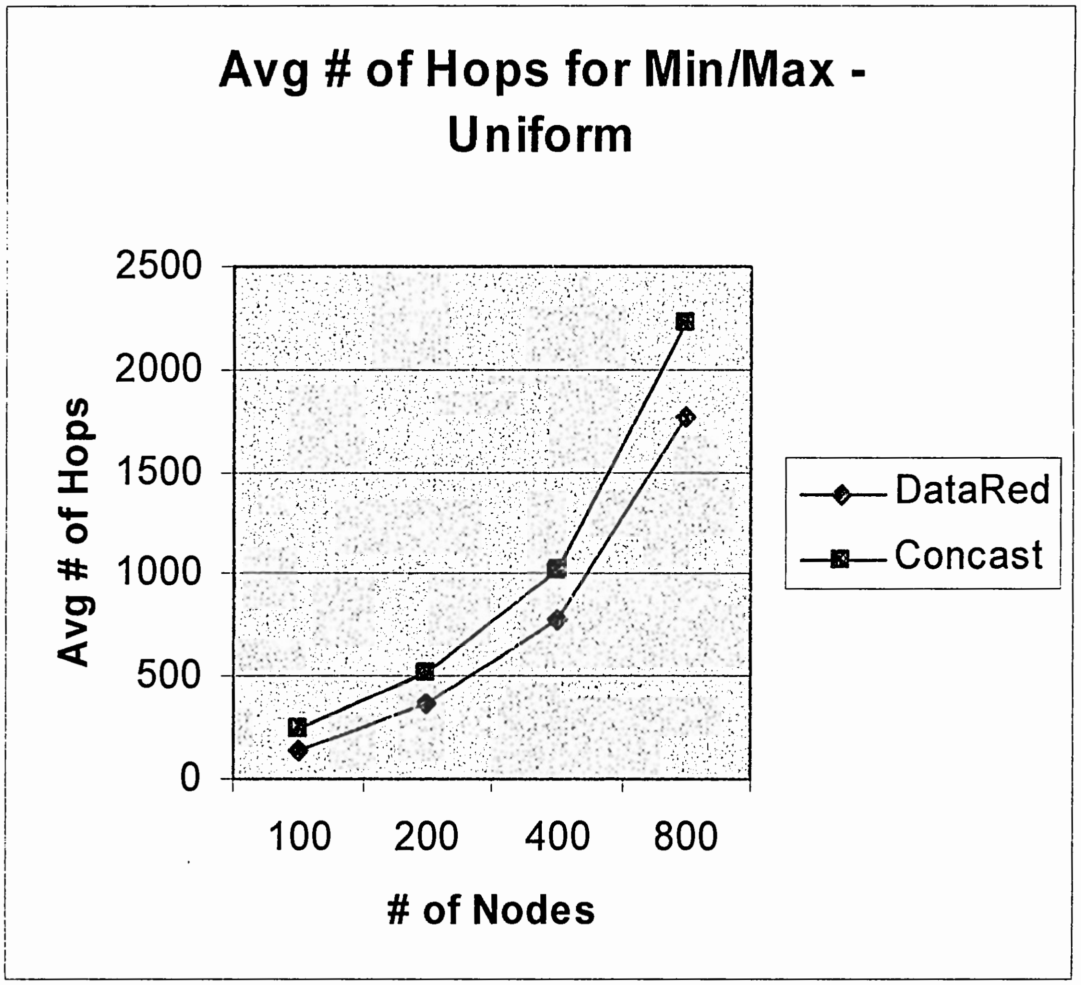

Figure 12: Avg. Number of Hops to Compute Min/Max with Uniform Node Dist.

Figure 12 shows the average number of Hops to perform min/max computation over a network using Concast and data reduction protocol. The set up was the same as described above. As can be seen in the graph, the data reduction protocol still completes the collection with lower number of hops than the Concast protocol. However, the difference between the performances of the two protocols is less than when computing sum. Concast achieves some improvement by keeping state at all the nodes and reducing the number of packets forwarded to the collector.

7.1.6 Amount of Time to Compute Min/Max

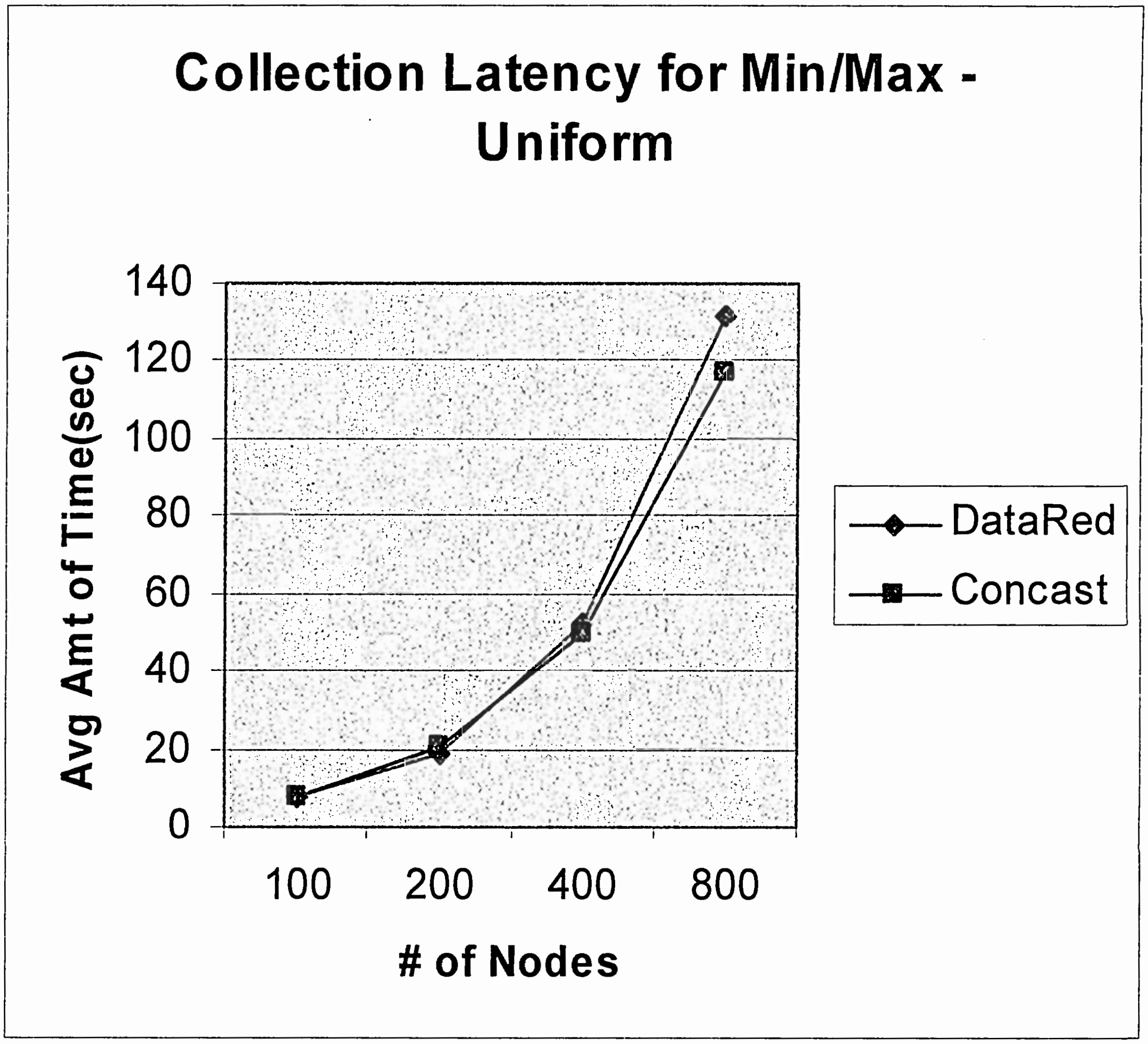

Figure 13: Amt. of Time Required to Compute Min/Max with Uniform Node Distrib.

Figure 13 shows the amount of time taken to perform the min/max computations. As can be seen from the graph, Concast protocol performs slightly worse than the data reduction protocol for small number of nodes. For large number of nodes, Concast surpasses the data reduction protocol. This is because of the larger number of children per parent in the Concast protocol for larger number of nodes. In the best case, a parent only has to report the value of its child node with the highest or the lowest value. This leads to a much larger number of packet reductions at parent nodes with larger number of children.

7.2 Performance Tests Over Bounded Normally Distributed Networks

Just as for previous section, the nodes were distributed over a 400 by 400 region with sixteen 100 by 100 predefined geographical regions. Three levels were used for the data reduction protocol just as in Section 7.1. The nodes in these tests were distributed over the entire area using bounded normal distribution as described in Chapter 5. The radio ranges were again set to 40. Some of the random placements produced unconnected networks. This was detected by noticing that a broadcast through the network did not reach all of the nodes. These networks were discarded and another random network was generated. Again an initialization stage consisting of n/2 messages was included. All the simulation results were generated by running third level collections over the entire network. The number of random networks tested was the same as in Section 7.1.

7.2.1 Number of Packets Required Per Collection

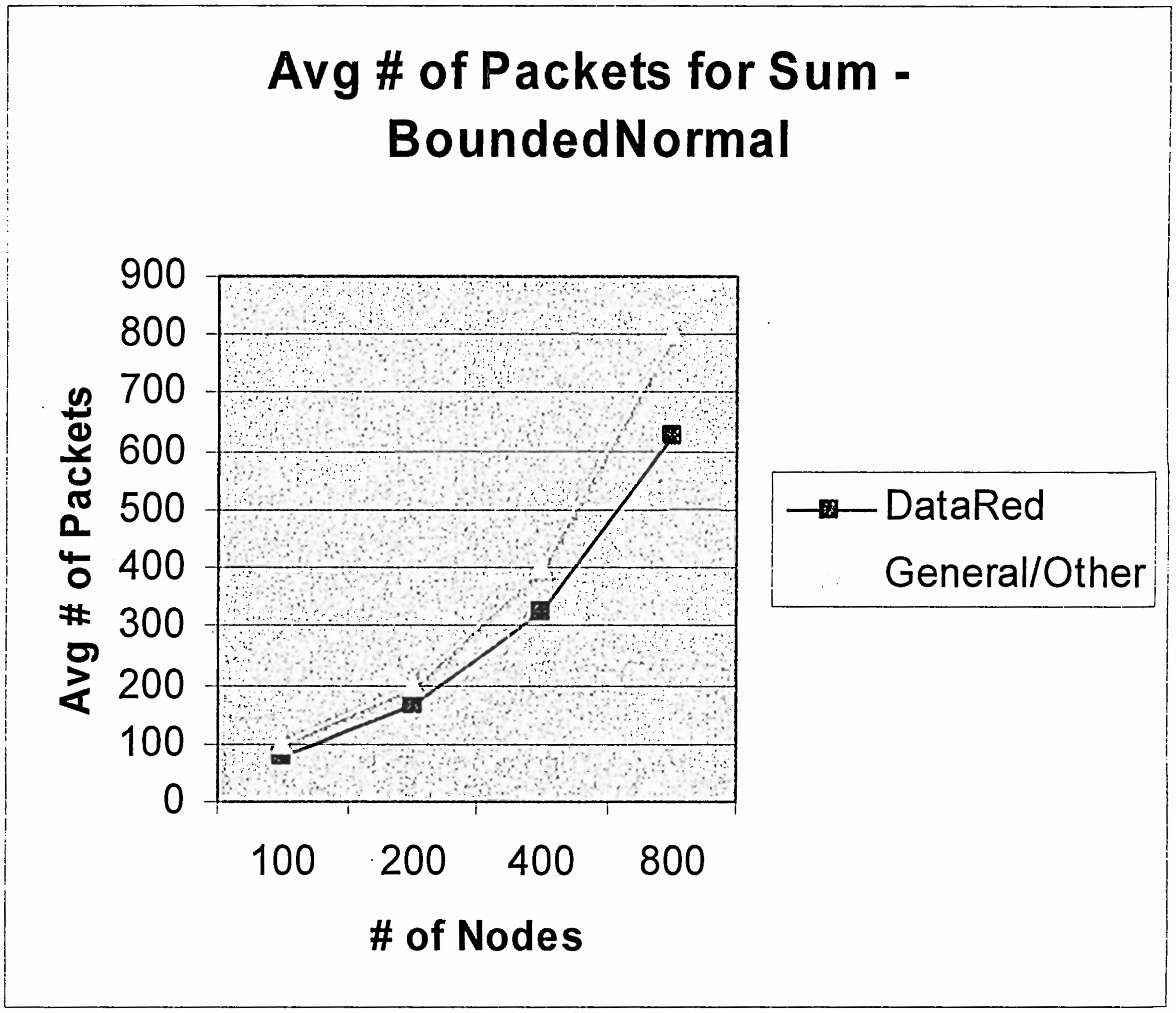

Figure 14: Avg Number of Packets per Collection with Bounded Normal Node Distrib.

Figure 14 shows the average number of packets required for a single collection for different number of nodes using Data Reduction protocol and the other three protocols. The data was collected by computing sum over random networks. The random networks were created as described above. For the general approach, Concast and TAG, the number of packets required is still equal to the number of nodes, because each node creates and transmits a single packet with its own data. Data reduction protocol still requires lower number of packets than the other protocols, but the difference is not as great as when the nodes were distributed uniformly. In this distribution, most of the nodes were clustered close to the center of the region. The collector node also has a higher probability of being close to the center. Because of this, a larger number of nodes are one or two hops away from the collector node. No reduction is done when a node is only one hop away from the collector. Therefore, the advantage in terms of packets for the data reduction protocol is not as much as in uniform distribution.

7.2.2 Number of Hops Required Per Collection

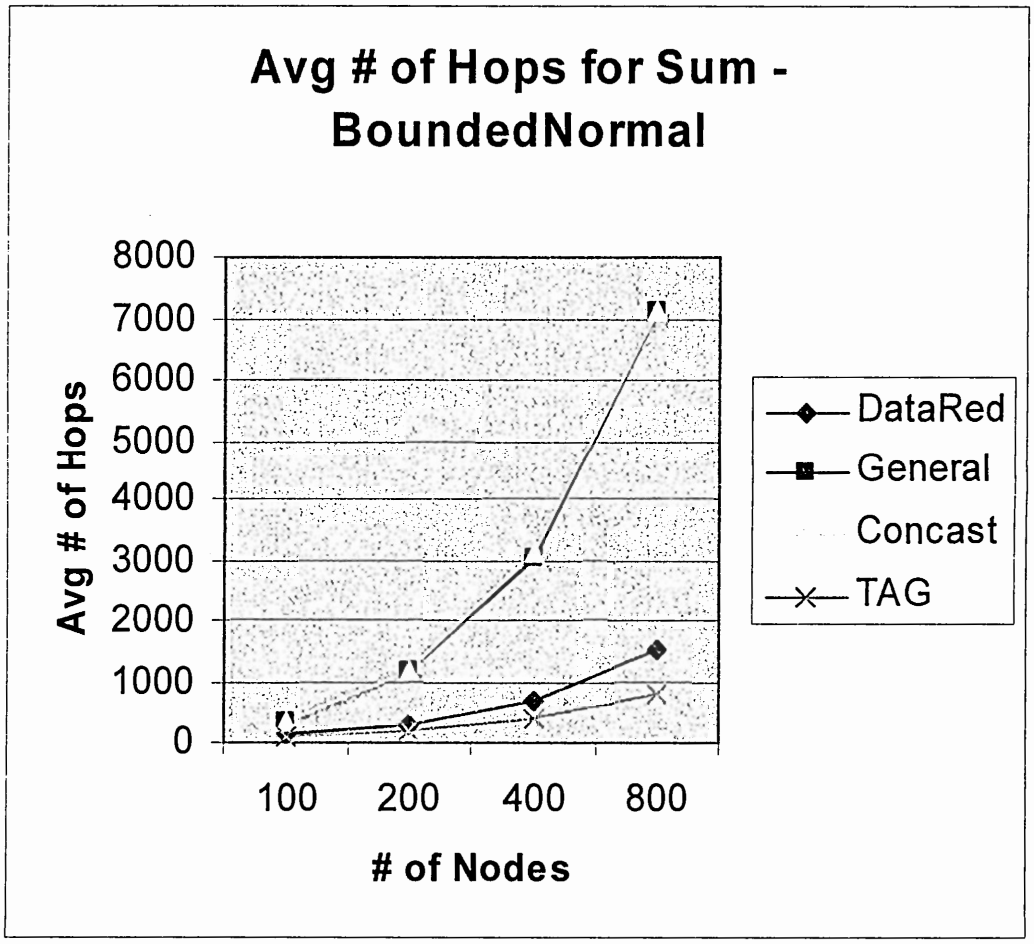

Figure 15: Avg Number of Hops per Collection with Bounded Normal Node Distrib.

Figure 15 shows the number of hops required for single collection. As before with the uniform distribution, the data reduction protocol still shows a significant advantage over the general approach. All the protocols except TAG took a lower number of hops per collection than in the uniformly distributed network. The reason is again the clustering of nodes at the center of the region. Collector near the center can reach more nodes with very few hops. There is a smaller chance of having a collector away from the center, since there is smaller number of nodes away from the center. The TAG protocol does not show any difference because it is already using the least number of hops possible. The Data Reduction protocol gets closer to the ideal represented by TAG in bounded normal distribution.

7.2.3 Amount of Time Required Per Collection

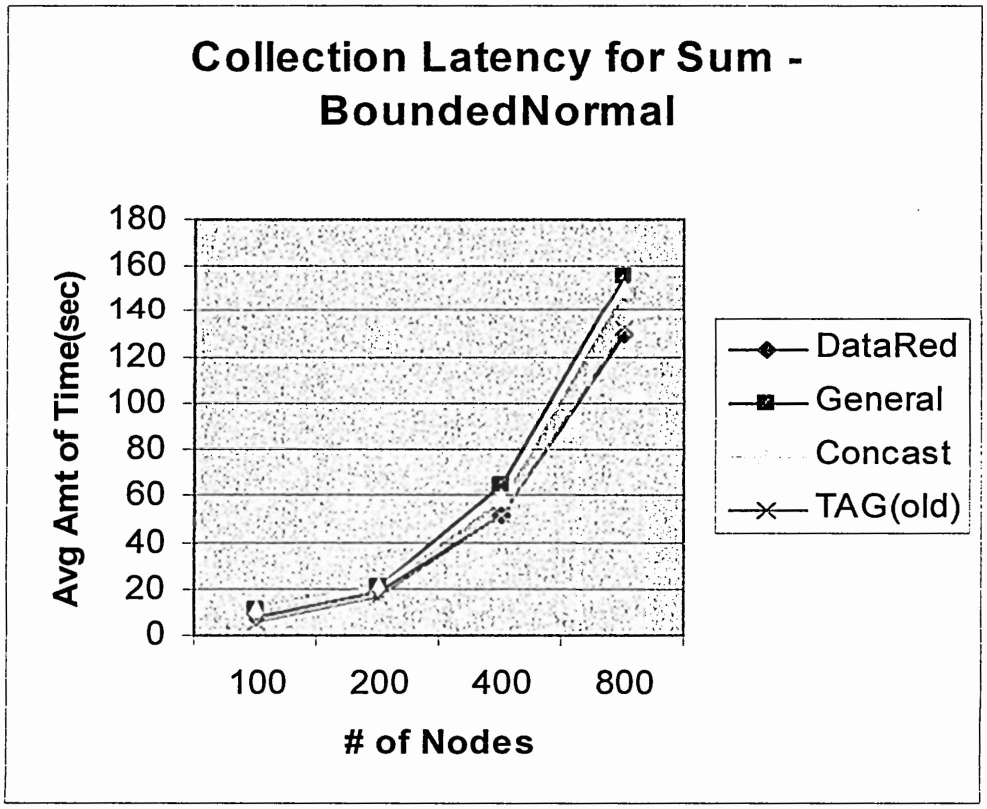

Figure 16: Avg Amount of Time per Collection with Bounded Normal Node Distrib.

Figure 16 shows the amount of time for single collection. As before, the data reduction protocol still has advantage over general and concast protocols, but the difference is much smaller in bounded normal distribution than in uniform distribution. All protocols take a little less time to complete one collection than in uniform distribution. The improvement is greater for the general approach. The reason is the same as one given in previous sections. For nodes one hop away from the collector, the general approach uses just as much transmissions as other protocols but does not require as much processing.

7.2.4 Number of Packets to Compute Min/Max

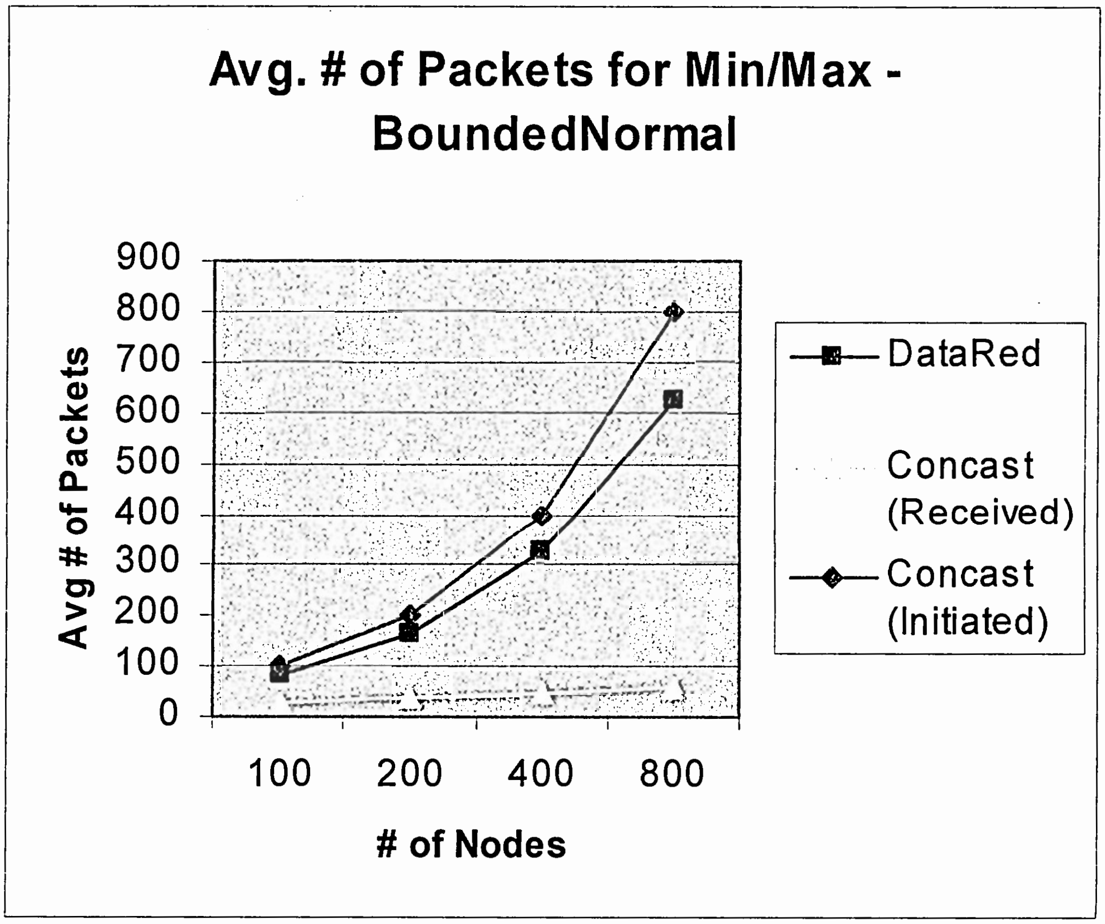

Figure 17: Avg Number of Packets to Compute Min/Max with Normal Node Dist.

Figure 17 shows the packets required to perform min and max operations. These operations were performed separately as with the experiment in uniformly distributed networks. Packets were dropped at intermediate nodes as before. The number packets received by the collector nodes in Concast is higher for bounded normal distribution. This is again because of the higher number of one hop neighbors to the collector.

7.2.5 Number of Hops to Compute Min/Max

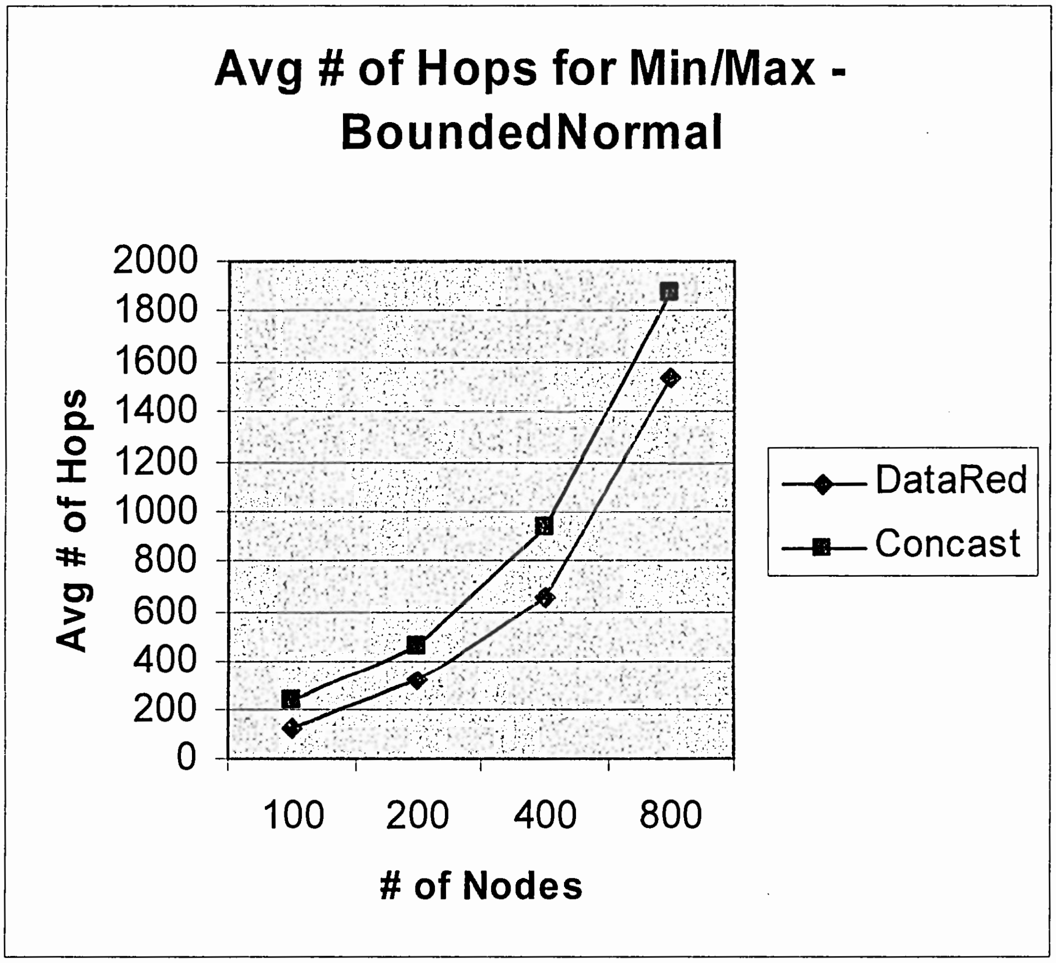

Figure 18: Avg Number of Packets to Compute Min/Max with Normal Node Dist.

Figure 18 shows the number of hops taken to perform the min/max operations. The data reduction protocol still holds advantage over the Concast protocol. Both protocols require fewer hops to compute min and max. The difference between the two protocols is about the same as in uniform distribution.

7.2.6 Amount of Time to Compute Min/Max

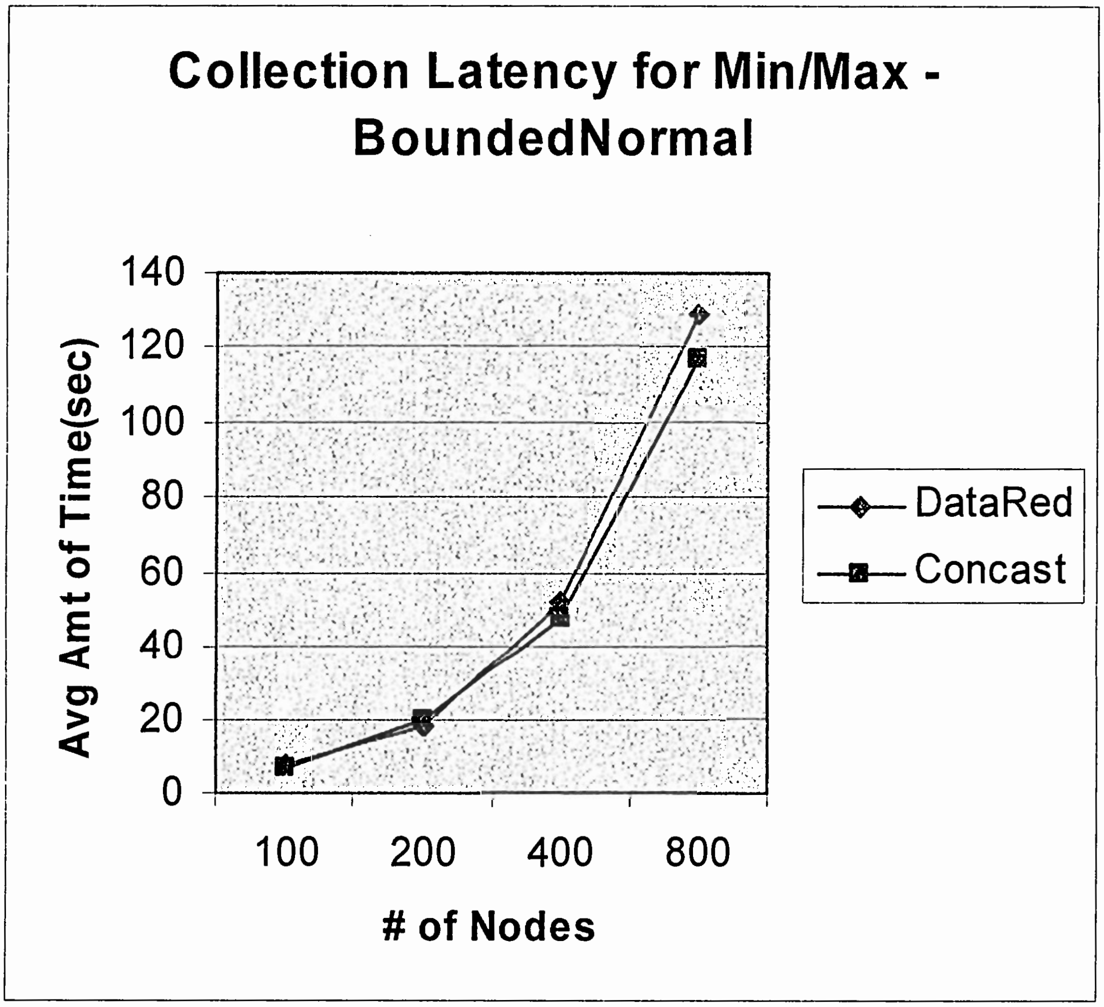

Figure 19: Amt of Time Required to Compute Min/Max with Normal Node Distrib.

Figure 19 shows the time taken to compute min and max values. The results were similar to results with uniform distribution. Both protocols took a little less time to compute the aggregates. The difference between the Concast and data reduction protocol at high number of nodes is a little less than the difference in the uniform distribution experiment. This is because most of the packet drops in the concast protocol occurred close to the collector rather than farther down the tree. This reduces the advantage of Concast over the data reduction protocol.

7.3 Performance Tests over Different Radio Ranges

As seen in Section 7.2, the advantage of the data reduction protocol over general approach was reduced a little for bounded normal distribution. This was hypothesized to be because of the increase in the number of one hop neighbors to the collector. To confirm this, data was collected on the effect of varying the radio range of all the nodes. Increasing the radio range while keeping the number of nodes fixed tends to increase the number of neighbors of each node, thereby allowing us to test this hypothesis. The total number of nodes was fixed to 200. The nodes were distributed over the same 400 by 400 regions using uniform distribution. The min operation was tested to check the performance of general, Concast and data reduction protocols.

7.3.1 Packets Required Per Min Operation

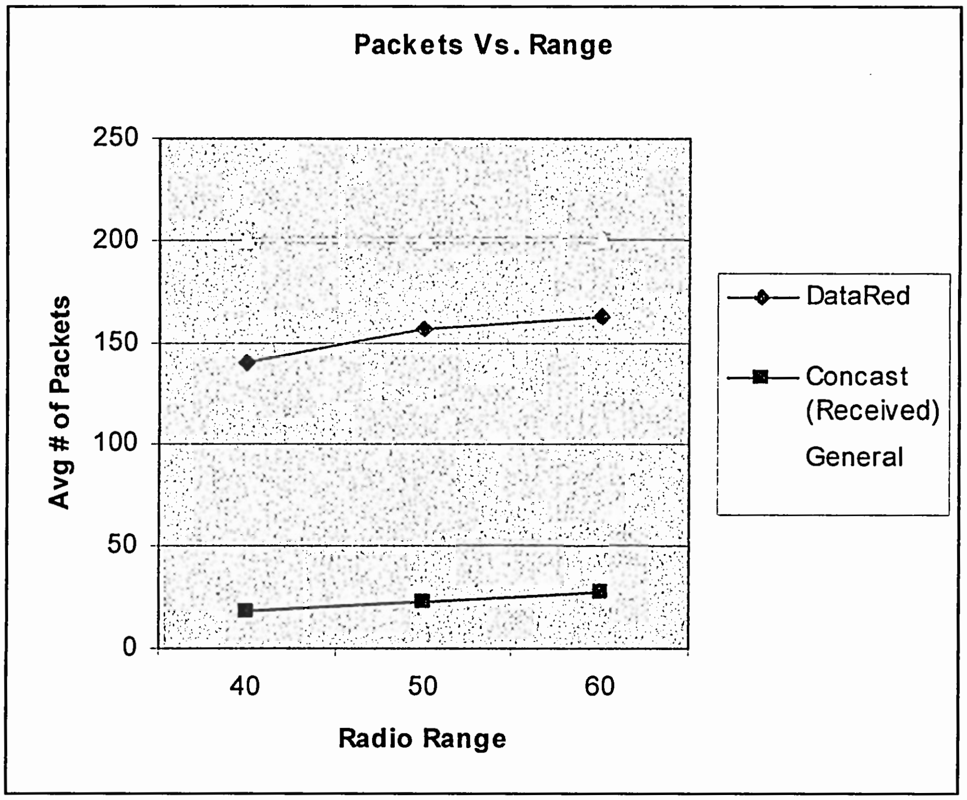

Figure 20: Avg # of Packets vs. Radio Range

Figure 20 shows the number of packets required to perform a single collection with different radio ranges. The general protocol requires the same number of packets for all radio ranges. For data reduction protocol, the number of packets required increases as the radio range increases. The number of packets received by the collector in the Concast protocol also increases with radio range. This is as predicted by the hypothesis.

7.3.2 Hops Required Per Min Operation

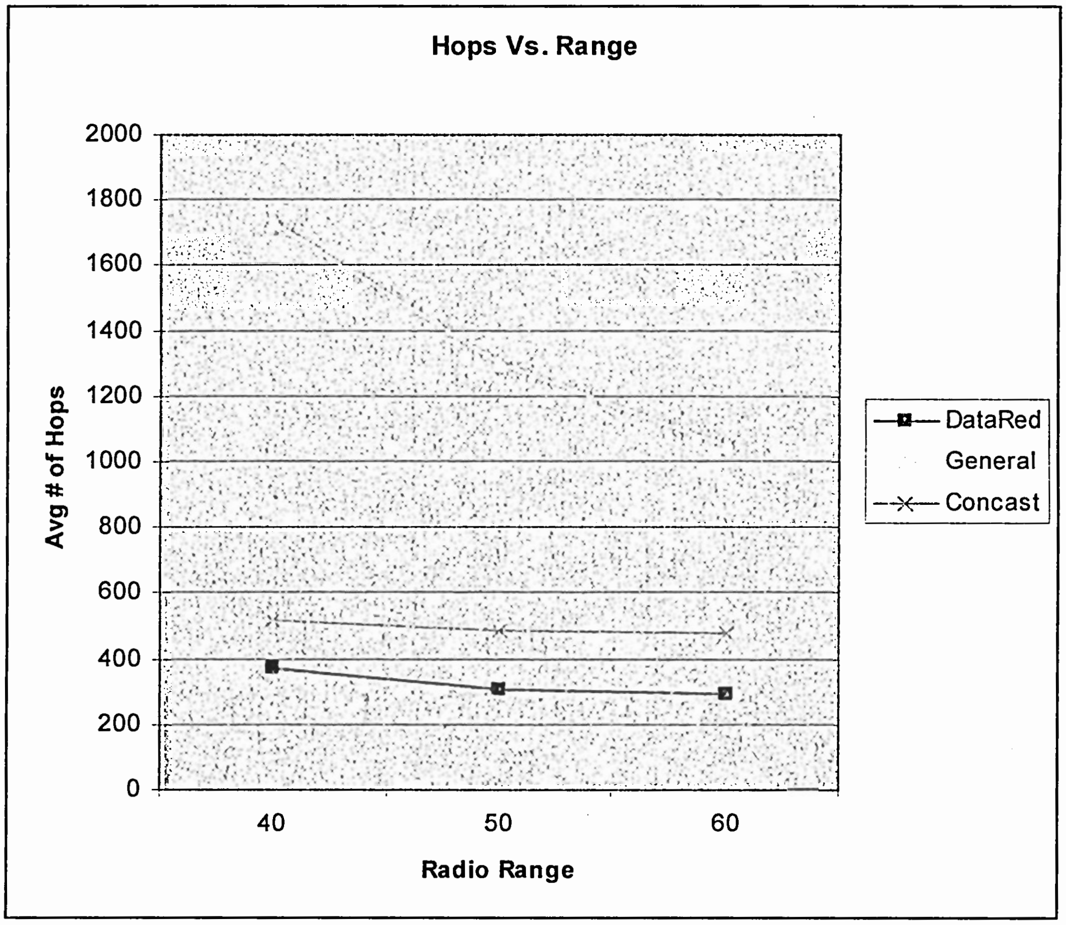

Figure 21: Avg # of Hops vs. Radio Range

Figure 21 shows the average number of hops required to perform a single min operation. The graph shows that all of the protocols require lower number of hops as the radio range of nodes increase. However, the drop is lot more significant in the general approach than in the data reduction and Concast protocol. This shows that as the range increases the number of reductions done due to the protocols decreases.

7.3.3 Amount of Time Required Per Min Operation

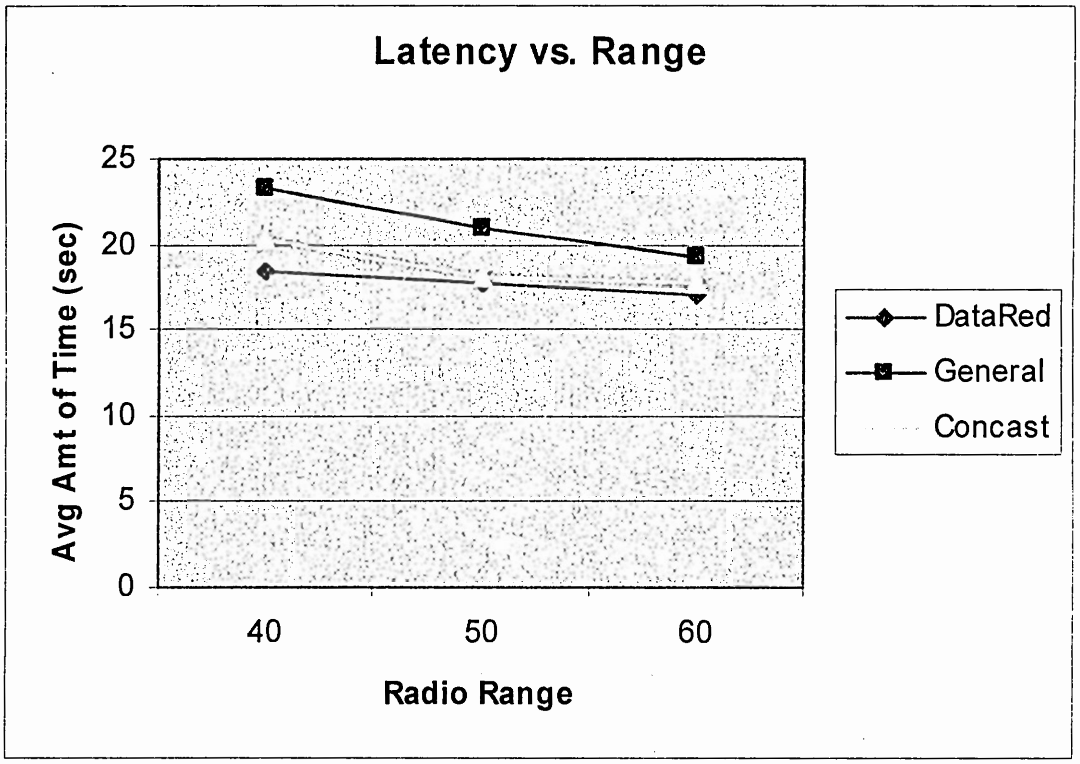

Figure 22: Avg Amount of Time vs. Radio Range

Figure 22 shows the effect of different radio ranges on the time required for min operations. Again the general approach benefits the most from the increase in radio range. The previous three graphs show that the data reduction protocol provides a larger benefit when a large number of nodes are not clustered around the collector. In the most extreme case if all the nodes are just one hop away from the collector, the general approach would give the best solution, provided that there is a good collision resolving mechanism. These graphs also show that the data reduction protocol is less vulnerable to reduction in radio range. If some nodes need to reduce their power levels to last longer, the effect on performance of the data reduction protocol would be small.

7.4 Performance Tests with Loss of Nodes

One of the advantages of the data reduction protocol over the TAG protocol is its performance and reliability when one or more nodes go down or move out of range. Tests were performed to measure the performance of these protocols with loss of one or more nodes. These tests were done on random networks generated using the same method as in Section 7.1. The count aggregate was collected. Each node was given a 1% chance of being turned off. A uniform random variable was used to decide whether a node should be on or off. The percentage difference between the actual result and collected result was measured. Latency was also measured for TAG and data reduction protocols. For data reduction protocol, DSR protocol takes care of off nodes. The node that precedes the off node in the spanning path would detect that the link is broken. This node sends an error message back to the collector. The collector node constructs new spanning paths to cover the nodes in the same spanning path as the off node and then sends requests on those paths. Nodes ignore the collection requests with the same request id and source as a previously received request. In TAG, the collection latency is highly dependent on the timeout values for the parent nodes in the collection tree. A different timeout value has to be used for different levels of the tree. Collections were done on ten randomly generated networks for each of the four different numbers of nodes. The max latency value from those ten collections was taken as the timeout value for the final collector for that number of nodes. The amount of time taken to collect data from one hop neighbors was taken to be the timeout for the second lowest level. The tree depth was also measured for each of the different number of nodes. The maximum value of the tree depth was assumed to be the depth for that number of nodes. The decrease in the timeout value was set equal to following equation:

Decrease at each Level = [(Timeout at Highest Level) – (Timeout at lowest level)] / depth

The following table shows the values used for different number of nodes:

Table 1: Timeout Values for TAG Protocol

# of Nodes	Max Timeout (sec)	Min Timeout (sec)	Depth
100	8.46	0.35	18
200	22.17	0.52	21
400	69.71	1.17	24
800	177.38	2.18	25

The following two graphs show the percentage error and collection latency for the TAG and data reduction protocols.

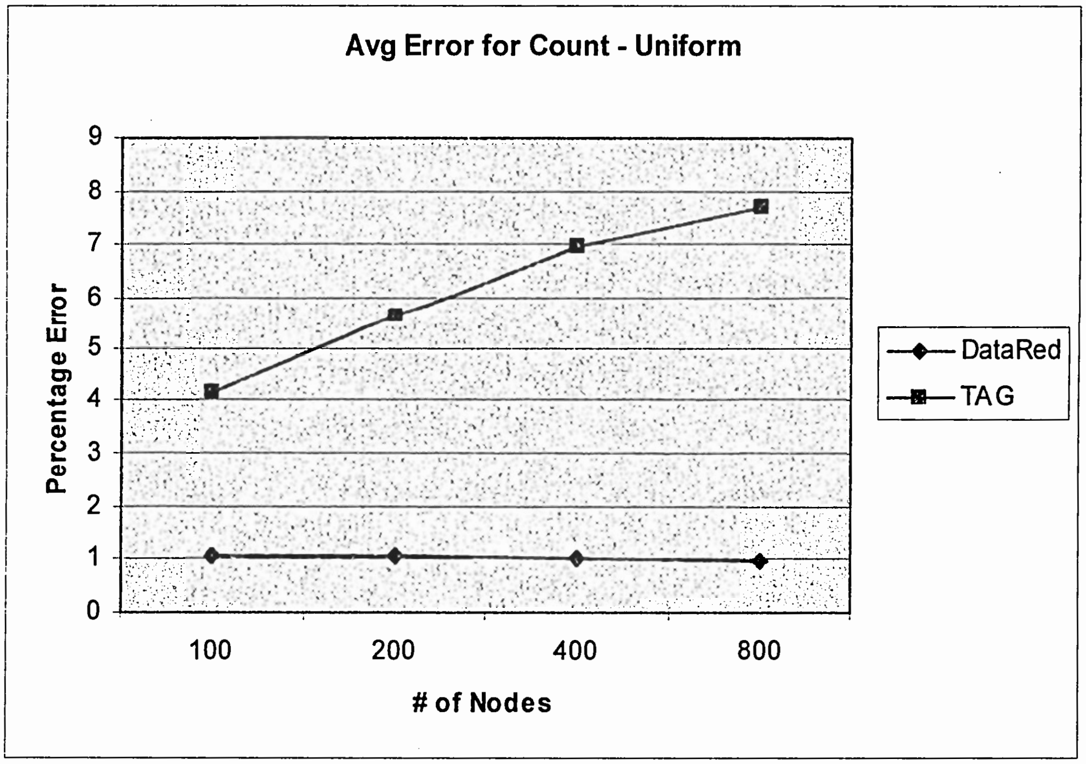

Figure 23: Avg. Percentage Error for Count with Loss of Nodes

When a node is turned off, TAG protocol looses the data from not only that node but also from every node below that node. This makes TAG less reliable than data reduction when nodes are lost. When computing count in data reduction protocol, the percentage of error is equal to the percentage of nodes that were turned off.

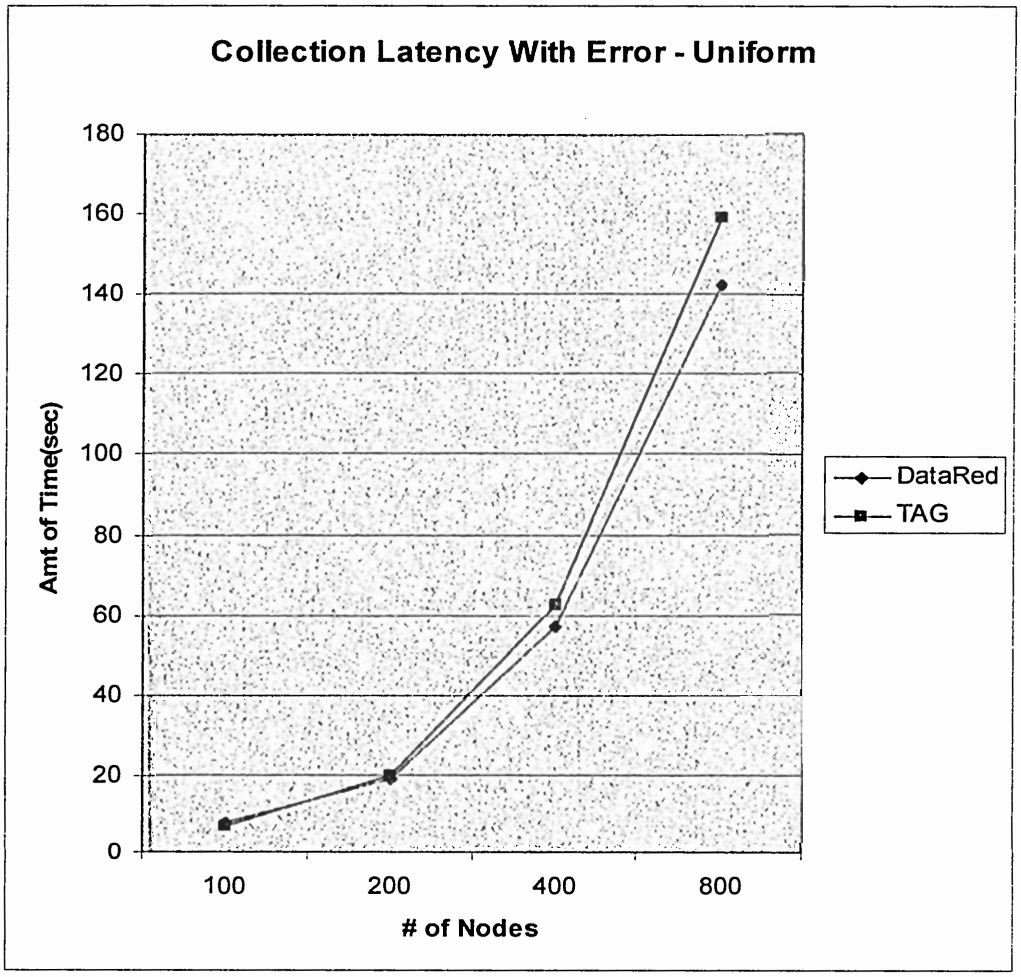

Figure 24: Avg Collection Latency for Count with Loss of Nodes

Figure 24 shows that the data reduction protocol outperforms the TAG protocol for large number of nodes. This is because the parent nodes in the TAG protocol have to wait until the timeout occurred before transmitting collected data. The data reduction protocol does not require any such waiting. Figure 23 and Figure 24 show that the data reduction protocol performs better than TAG when there is node loss.

CHAPTER VIII

CONCLUSION

8.1 Summary of Results

The goal of this thesis was to design and evaluate a new protocol for data collection in a sensor network. Such a protocol would be useful in many wireless sensor applications where periodic data collection is a frequent operation. The goal was to design a protocol that minimizes the amount of transmissions and hence the amount of energy used for each collection. The protocol also had to be simple and as stateless as possible to conserve the limited resources of the sensor nodes. Since most future applications of sensor networks consist of a very large number of small sensor nodes, the protocol also had to be scalable. The data reduction protocol accomplishes these goals. It is simple and requires very little processing and memory at the nodes. Only the few collector nodes are required to keep state information for each ongoing collection. In a one-level collection, only the request originator needs to maintain state. This is an improvement over the other current protocols. Both TAG and Concast protocols require each node to keep state information about the identity of its parent and current state of collection. The data reduction protocol requires almost four times lower number of hops than the general approach. The difference becomes larger as the number of nodes increases. This shows that the protocol is more scalable then general approach. Data reduction protocol also requires lower number of hops than the Concast protocol even when computing min/max.

The number of packets created in the data reduction protocol is lower than in any of the other protocols. This protocol also shows a slight advantage in the amount of time taken to complete a collection. The advantages are there for both uniform and bounded normal distributions. The performance of data reduction protocol does not fall as sharply as the performance of general approach when the power level, hence the radio range, of the nodes is lowered.

Another advantage of data reduction protocol over TAG and Concast is that it could be implemented on top of any existing source routing protocol. The error correction and reliability of a well developed source routing protocol such as DSR can be used to ensure reliability of the data reduction protocol. This makes data reduction more robust for ad hoc mobile networks. The spanning routes are generated by the collectors each time on demand, although they can also be cached for efficiency. DSR handles mobility by removing routes that are no longer valid and discovering new routes if necessary. Since the DSR routes are used to generate the spanning paths or to update stored spanning paths, data reduction protocol can also handle mobility. In TAG and Concast protocol, the collection tree has to be rebuilt periodically to account for mobility. This could be time consuming, and in between periodic rebuilds, the protocols will give false results if mobility occurs.

Even though TAG protocol requires less hops then the data reduction protocol, it requires more state and processing. The parent nodes have to remember the number of children and have to set a timer to wait for all child nodes to report. This has to be done at all nodes except at few leaf nodes with no children. TAG is also less reliable, especially when there is mobility. Data reduction protocol outperforms the TAG protocol when there is loss of nodes. The data reduction protocol becomes more efficient as more routes become known through DSR. After a while, route discoveries do not have to be done to find spanning paths. In TAG protocol, performance is not improved as more messages are exchanged.

The above analysis and results have shown that data reduction is a viable alternative to existing collection protocols for many practical applications of sensor networks. The protocol can be modified to perform better in certain situations. For example if amount of energy used is a very critical condition, nodes that fall on more than one spanning paths can try to wait and combine the results from the two spanning paths. This would make the protocol similar to TAG protocol and reduce the number of hops. Next section discusses some other optimizations to be explored for the data reduction protocol.

8.2 Future Work

The multi level data reduction protocol requires the network region to be divided into smaller regions and each node knowing the number of its local region. More research needs to be done on how to accomplish this task. The geographical boundaries of the regions can be pre-determined by user based on the network topology. Each node can be made aware of these boundaries when the network is created and can calculate its current region based on its location. There have been some papers on how a node can determine its current location based on distances relative to some other fixed nodes [16, 17, 18]. These protocols have to be analyzed and evaluated for use with data reduction protocol.

One essential process in the data reduction protocol is the selection of collector nodes for lower levels. The collector node is currently chosen by simply selecting one node per region that is in one of the spanning routes. More than one node from some of the regions can be in one or more spanning routes. The current protocol arbitrarily selects a collector node for that region. More work needs to be done on this collector selection process. The collector can be chosen based on its power level, its position or some other criteria. Tests have to be conducted to see which selection algorithm gives the best results. Since the collector nodes are required to do additional processing, rotating of collectors could prolong the life of the network.

The test results presented in this paper were obtained by running the protocol on the GTNetS simulator. Although the simulation environment reflects the real world applications, the protocol has to be tested on actual sensor nodes in real world applications. The power requirements of the protocol have to be studied further to evaluate the benefit of such optimizations as keeping the sensors off until receiving a collection request.

REFERENCES

[1] A. M. Law and W. D. Kelton. "Simulation Modeling and Analysis, 3rd edition" pg. 465-466, McGraw-Hill Companies, October, 1999.

[2] D. Johnson, D. Maltz, and J. Broch, "Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks," in *Ad Hoc Networking* (C. E. Perkins, ed.), ch. 5, pp. 139–172, Addison-Wesley, 2001.

[3] C. E. Perkins and E. M. Royer, "Ad hoc on-demand distance vector routing." in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90–100, Feb 1999.

[4] D. M. Blough and Andrzej Pelc. "A Clustered Failure Model for the Memory Array Reconfiguration Problem." in *IEEE Transactions on Computers, Vol 42.* pp. 518-528, May 1993.

[5] D. M. Blough and G. F. Riley. "ScalableSensorSim – A Simulation Enviroment for Large-Scale Sensor Networks." Proposal for NSF.

[6] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, J. Anderson. "Wireless Sensor Networks for Habitat Monitoring." Proceedings of WSNA, September, 2002.

[7] K. L. Calvert, J. Griffioen, B. Mullins, A. Sehgal, S. Wen. "Implementing a Concast Service." Proceedings of the 37th Annual Allerton Conference on Communication, Control, and Computing. September, 1999.

[8] G.J. Pottie, W.J. Kaiser, "Wireless Integrated Network Sensors," *Communication of the ACM,* vol. 43, no. 5, pp. 551-8, May 2000.

[9] J. M. Kahn, R. H. Katz and K.S. J. Pister, "Mobile Networking for Smart Dust." ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom 99), Seattle, WA. August 17-19, 1999.

[10] J. Warrior, "Smart Sensor Networks of the Future," Sensors Magazine, March 1997.

[11] R. Collin Johnson, "Companies test prototype wireless-sensor nets." EE Times Magazine. January, 2003

[12] B. Krishnamachari, D. Estrin, S. Wicker, "Modelling Data-Centric Routing in Wireless Sensor Networks," IEEE Infocom 2002.

[13] J. Hill, R. Szewcxyk, A. Woo, S. Hollar, and D.C. K. Pister, "System Architecture directions of networked sensors." ASPLOS 2000, Cambridge, November 2000.

[14] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TAG: a Tiny Aggregation Service for Ad-hoc Sensor Networks", OSDI 2002, December 2002.

[15] C. Intanagonwiwat, R. Govindan, and D. Estrin. "Directed diffusion: A scalable and robust communication paradigm for sensor networks." In MobiCOM, August, 2000.

[16] D. Niculescu, B. Natth. "Ad Hoc Positioning System (APS) ". In Proceedings of IEEE GLOBECOM '01, November 2001.

[17] A. Nasipuri, K. Li, "A Directionality based Location Discovery Scheme for Wireless Sensor Networks", WSNA 2002, September 2002.

[18] S. Capkun, M. Hamdi and J.P. Hubaux, "GPS-free positioning in mobile Ad-Hoc networks," Hawaii International Conference On System Sciences, HICSS-34, January 2001.