SCALABILITY AND COMMUNICATION WITHIN SWARM COMPUTING

A Thesis
in TCC 402

Presented to
The Faculty of the
School of Engineering and Applied Science
University of Virginia

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Computer Science

by

Ankush Seth

March 24, 2003

On my honor as a University student, on this assignment I have neither given nor received unauthorized aid as defined by the Honor guidelines for Papers in TCC Courses.

_____
Ankush Seth

Approved _____
Dave Evans – Technical Advisor

Approved _____
Rosanne Simeone – TCC Advisor

Table of Contents

Table of Figures

Glossary of Terms

Dispersion: It is the movement of swarm devices from a clustered or localized arrangement to one that is uniform in distribution with respect to the specified terrain.

Scalability: The property that allows a system to expand with minimal loss in performance or functionality of the system.

Swarm Computing: A method of computing that involves a large number of small independent devices that communicate with each other to perform an assigned task.

Terrain: The region within which the swarm devices work.

Abstract

Consider the potential application of space exploration that uses mobile swarm devices. The feasibility of such an application depends on the ability of the devices to disperse uniformly, when starting from a clustered position, and on the scalability of the protocol used for dispersion. This research project looked at the scalability and communication aspects of the disperse primitive so as to come up with protocols that will help make such applications possible.

Swarm computing is a field that involves a large number of small independent devices that communicate with each other to perform an assigned task. Scalability is the property that allows a system to expand with minimal loss in performance or functionality of the system, whereas the disperse primitive enables the swarm devices to move from an initially clumped position to one that is dispersed.

Three protocols were developed and these were the random protocol, random with history protocol, and the coordinated protocol. The evaluation of the protocols showed that the coordinated protocol was the best in terms of performance measures like time, energy, and the number of steps taken to achieve the good state.

1.0 Introduction

Swarm computing is an idea fostered by the advent of faster chips, miniaturization of devices, and the evolution of distributed computing techniques. Conceptually speaking, swarm computing involves a large number of small independent devices that communicate with each other to perform an assigned task. It is a field that has hundreds of potential applications, from vacuuming houses to monitoring of hostile terrains. This technical report primarily focuses on the scalability and communication aspects of the disperse primitive. Scalability is the property that allows a system to expand with minimal loss in performance or functionality of the system, and the disperse primitive enables the devices to move from an initial clumped position into one that is dispersed. The scalability of the disperse primitive was analyzed by, simulating disperse protocols that were developed as part of the project. The results showed that the coordinated protocol was the best as it scaled linearly with respect to time, energy, and the per unit number of steps taken, as the number of devices in the system was increased. On the other hand the random and random with history protocols scaled exponentially.

1.1 Swarm Computing

Swarm computing involves numerous devices that work in a coordinated fashion, communicating either with neighboring devices or the environment to accomplish a defined task. It is a concept analogous to swarms that exist in nature, such as swarms of bees that also work collectively to achieve a common goal. There is no centralized server controlling these devices. Moreover, these devices have low computational and communication capabilities, and thus are limited to applications that can be accomplished

without extensive communication and information collection. For example, swarm devices could be used for space exploration. Such an application requires minimal interaction with the neighboring devices so as to explore the given terrain.

Swarm computing as a concept also draws upon research in fields such as Amorphous computing that deal with a system of irregularly placed asynchronous, locally interacting computing elements (Homsy, Knight and Nagpal 1). Each of these elements, like swarm devices, have modest computing power and are programmed identically (Homsy, Knight and Nagpal 1). These devices communicate via short-distance radio or by chemical means. Similarly swarm devices use radio communication to interact with each other.

Swarm computing made its appearance when Chris Langton founded the swarm project in 1994 at the Sante Fe Institute. Since then swarm computing has come a long way. People like Paul Edward Johnson have played an important role in the creation of the Swarm Development Group. The challenge today is to be able to develop swarm programs based on disparate primitives like dispersion, scaling, etc. especially, since the hardware necessary to implement swarm computing has improved by leaps and bounds in the past few years. Micro devices and wireless networks of various kinds have been built to provide the infrastructure necessary for swarm computing (Evans 11). But before the development of such programs takes place, extensive research is required on the behaviors and properties that need to be incorporated as part of the swarm programs.

1.2 Scalability and Communication

Scalability is a property that is essential to the successful performance of any system that involves large numbers of devices, as in the case of swarm computing. For example, the number of devices needed to monitor a hostile terrain using swarm devices would depend on the size of the terrain, and one would want that the application performs well regardless of the number of devices in the system. Thus a system that is scalable will allow it to expand with minimal loss in the performance or functionality of the system. In other words, the Big-O value of a scalable system should remain the same.

The main goal of this research is to look at the scalability and communication aspects of the disperse primitive within swarm computing. Though research on the disperse primitive exists, scalability of the system has not been analyzed in great detail. For example, Michael Cuvelier, looked at algorithms such as the Random Algorithm and the 0-Threshold Algorithm, to determine the most efficient one of them. Scalability was not one of his performance measures. He only did fixed size experiments. This project on the other hand looks at how various factors like initial position of the swarm devices, strength of the radio transmission, affect the scalability of the protocols used by the swarm devices. The project analyzes how factors like energy consumption per device, and the number of steps taken by each device, scale with an increase in the number of swarm devices present within a given terrain. Energy consumption includes both, the energy required for message transmission between the devices and the energy required for the movement of the devices. The number of steps basically represents the total distance a given device needs to move in order to reach a dispersed state. Besides scalability, communication was also a pertinent factor in this research. Radio

communication was the only way the swarm devices could collect information about their environment or neighboring devices. Therefore different protocols were developed each involving different amounts of information being broadcasted using radio communication.

1.3 Disperse Primitive

Dispersion forms an important aspect of swarm computing when mobile devices are involved. Dispersion is the movement of the swarm devices from a clustered or localized arrangement to one that is uniform in distribution with respect to their specified terrain. For example if swarm devices were used to monitor a hostile terrain, the disperse primitive would allow the devices to spread uniformly across the terrain, once they've been placed at a certain location within the terrain. Automating dispersal would save the armed forces or whoever is using the devices valuable time, as they themselves would not have to place the devices all over the terrain. An important factor relating to dispersion is that the dispersion protocol should enable the devices to reach the dispersed state quickly. The protocol should also scale well, that is, as the numbers of devices are increased the time taken for the devices to achieve the dispersed state should be increasingly linear with respect to the performance measures that have been set for the protocol, and not exponential.

Figure 1.1 and figure 1.2 illustrate this primitive. The first figure shows nine devices (portrayed as blue squares) in a clump at one corner of the terrain (the black box) and the second one shows these devices to be spread in an approximately uniform

configuration. In reality the number of devices will probably range from tens to few thousands of devices, depending on the application they undertake.

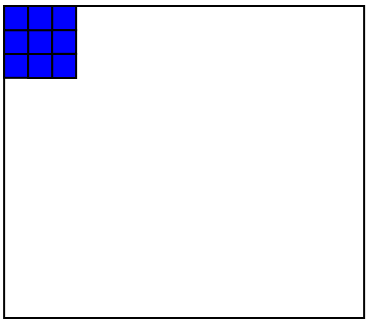

Figure1: Swarm devices in a clustered configuration.

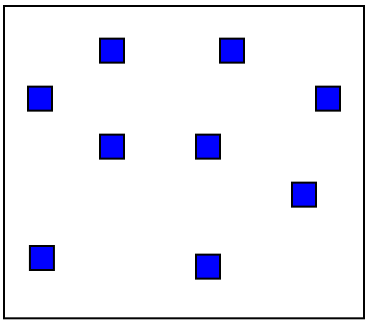

Figure2: Swarm devices in a uniform configuration

1.4 GloMoSim

GloMoSim, Global Mobile Information Systems Simulation Library, is the simulator used for this project (Jay 1). Developed at the University of California, Los Angeles, GloMoSim simulates devices connected via wireless network protocols. GloMoSim allows scaling through parallel execution of simulated agents and in our case these agents are swarm devices (Jay 2). GloMoSim implements various radio communication protocols and also provides the functionality of simulating user written applications.

GloMoSim has a layered structure and the whole simulator has been built such that each layer handles one aspect of the simulation. This project basically deals with the development of code for the top layer that is the Application Processing layer.

1.5 Materials and Methods

This section describes the design and development methodologies employed for the implementation of the various parts of the research project. Being a computer science related research project, no special materials were required. Development basically consisted of computer software coding.

The code developed implements the disperse primitive for the GloMoSim simulator. Since the simulator was developed in C, the same language was employed in writing the code. The standard approach of a header and an implementation file was used. The header file defines the various function definitions whereas the implementation file contains the actual code for the functions defined in the header file.

The software developed implements the algorithm representing the disperse model. It also encodes calculations that measure things like power consumption and the time spent in achieving various constraints. Basic software engineering principles such as the commenting of code, proper indentation, avoiding variables that are not initialized were followed.

The graphs and results presented in this report were created using MS Excel. The disperse application was developed so that it generates its statistics files in the comma separated values (CSV) format. These files can be directly opened using MS Excel and this makes it easy for the user to generate graphs and do other calculations on the imported values.

1.6 Document Overview

This section provides a brief overview of what other chapters of this technical report cover. Chapter 2 describes the Disperse application and discusses the various protocols that were developed. Chapter 3 discusses the simulations that were run to test the protocols. This chapter also presents the results of the various simulations. Chapter 4 presents the conclusions, by providing a brief summary, an interpretation, and recommendations for possible future work.

2.0 The Disperse Application

The Disperse application implements the code for the disperse primitive, and integrates it with GloMoSim. The application was developed by following an approach similar to the one used by the developers of GloMoSim. Applications in GloMoSim follow a client/server model. The client and server interact via GloMoSim's wireless network protocol. The simulator executes the appropriate code depending upon the message being passed. The following sections focus at the development of the disperse protocol and not so much at the framework that integrated the protocol with the simulator.

2.1 Assumptions

This project makes a number of assumptions about the swarm devices and its environment, and these have been listed below:-

1)Terrain and Location: It is assumed that the terrain within which the swarm devices act is square shaped, and that the swarm devices do not know anything about their location within the terrain. It has also been assumed that there is some sort of mechanism that allow the devices to figure out if they are within the terrain or not.

The reason for such an assumption is that swarm devices are small and have low computational capabilities and finding the coordinates within the terrain would require quite a bit of processing and interaction with the either the environment or other devices. This assumption also limits the types of protocols one can develop for dispersion.

2) Memory: It is assumed that memory is not an issue and that the swarm devices can store a decent amount of information.

3) Movement and Energy consumption: Another important assumption is that the devices move using a motor and gearbox system similar to radio controlled toy cars. Specifically, I assumed that the swarm devices use 3 volt motors that have a specification of 7300 RPM (Revolutions Per Minute) at 117mA (Direct Drive 1). Hence, the devices run on two standard AA batteries that have an average life of 2850mAh (Battery Specialist 1). The energy consumption during the movement of the device is given by the product of 351mW and the number of seconds the device moves. This number is an underestimate because it doesn't take into account the energy consumed by other electronic parts used in the movement circuit. However, the energy consumed by parts other than the motor would be negligible. Also, the energy required to transmit a single message is given by the following equation:-

$$\text{Energy} = \frac{\text{size of data (Bytes)} * \text{Energy required to transmit the message (10mW)}}{\text{Bandwidth of the communication signal (Bytes/s)}}$$

4) Speed: This assumption relates to the speed at which the swarm devices can move. It is assumed that the swarm devices can achieve a maximum speed of 10m/s depending upon the gear ratio used. The Disperse application has been set with an interval period of 5 seconds so as to allow the devices to reach their scheduled positions, before the next positions are calculated.

5) Neighbor range: It is assumed that the neighbor range for each device is 50 meters. A given swarm device is considered to be a neighbor of another device if it is located within the neighbor range of the other device.

6) Dispersed: A device is considered to be dispersed if it has four or fewer neighbors.

2.2 Protocols

The goal of this research project was to come up with a protocol that was scalable with respect to time, movement, and energy. Achieving this would have been relatively easier, if each node knew everything about every other node in the system and, also knew about its position relative to others. Three protocols were developed as part of this project, and these were:-

- The Random Protocol
- The Random with History Protocol
- The Coordinated Protocol

1) Random Protocol: In this protocol each node only knows about the number of neighbors it has at a given point of time. The swarm devices don't store any sort of information for later use. Each node keeps track of the number of neighbors by maintaining a running sum of the number of messages received. Each message represents a neighbor. The node also stores the time stamp of when the message was received. The protocol updates the 'number of neighbors' variable and the time stamps, if more than 5 seconds have elapsed from the time any given neighbor was added to the list.

The decision of whether the device should move or not depends on the number of neighbors it has at that point of time. If the number of neighbors is greater or equal to three, then a random direction and speed is chosen. The speed chosen is bounded by the maximum speed that the device can move at. Based on the information calculated, the

device moves a certain distance in the next 5 seconds before the next set of messages arrive. Therefore the movement calculations are done in intervals of 5 seconds.

2) Random with history protocol: This protocol also has the same structure as the one above but it maintains a history of the number of neighbors and the direction the node took in the previous step. If the current number of neighbors is less than the number of the neighbors the device had in the previous step then the device maintains the same direction. If not, then the device waits till the next set of messages, and chooses a new direction for moving. The node waits for one time interval before choosing a new direction so as to decrease the probability that all the nodes are moving in the same direction at the same time.

3) Coordinated protocol: This protocol expands on the features of the second one. In addition to keeping track of the direction, each node also keeps track of the number of steps it has taken. A step basically represents the movement of the device during the 5 second interval. Each node in this protocol broadcasts as part of its message the number of steps it is going to take. This value is initialized to '-1' for each node. When the nodes start receiving the messages if the value for the steps variable is '-1' it sets a new value by picking a random number. This is broadcasted as part of the new messages sent out by the nodes, and is used to pick values for the step variable that are different from its neighbors. Once a step value has been set it is not changed until the node has moved the specified number of steps and, therefore there is counter to keep track of the number of steps taken. Once completed, the steps variable is again set to its initial value of '-1'.

Besides the above mentioned features the coordinated protocol also has a variable that limits the choice of direction. For example if a certain direction results in the device moving out of the terrain, the device will not take that direction from the next time.

3.0 Evaluation

This chapter evaluates the performance of the protocols that were presented in the previous chapter. The chapter also presents information about the settings and performance measures that were used during the simulations. A mathematical model calculating the optimal values for the per unit number of steps required to reach the good state, has also been presented in this chapter.

3.1 Simulations

The simulations were run on GloMoSim, a wireless network protocol based, application simulator. Each protocol was simulated several times, so as to obtain maximum information about it. Two sets of simulations were performed using each protocol. The first set involved simulations in which, the devices were initially placed, as a clump, at the center of the terrain. Also, the number of nodes that were simulated was increased with each simulation. The second set of simulations was similar to the first one in every respect, except that the devices were initially placed in one corner of the terrain.

3.1.1 GloMoSim Settings

GloMoSim allows users to set various parameters relating to the simulation environment. These parameters deal with things like radio type, strength of the radio signals, and bandwidth. Most of the default settings were used, although, the 'radio-tx-power' variable was set to 10 dBm. This is because, after a few simulations it was observed that the protocols performed better at this radio signal strength.

3.1.2 Performance Measures

The performance of each protocol was measured in terms of certain factors, and these have been described below:-

- Energy: Energy consumption is an important issue because the swarm devices are battery operated, and for them to work for a long period of time, battery consumption should be as minimal as possible. Since the energy consumed is dependent on the movement of the devices, the fewer the number of steps taken by a given device, the lower is its battery consumption.
- Good State: Good state refers to the percentage of swarm devices that have reached a given level of dispersion. Each simulation is set with a certain good state value that acts as the minimum value for the dispersion and thus the devices are considered to be in the disperse state if the system achieves that level. For example a good state condition of 65% with four or less neighbors would mean that at least 65% of the devices in the simulation must have four or fewer neighbors for the system to be considered dispersed.
- Time: Time is a prime factor as it is important to know whether a given method of dispersion is linearly or exponentially dependent on time.

3.2 Results

The results have been provided in the form of different graphs, and these have been categorized on the basis of the initial configuration of the swarm devices within the terrain. Assuming the terrain is a square, the devices could either be placed in one corner

of the terrain (corner configuration) or they could be placed at the center of the terrain (center configuration).

3.2.1 Center Configuration

In this configuration the devices are initially placed at the center of the terrain, and therefore it is the easy for the devices to disperse from this configuration, as they have the complete 360 degrees to choose a direction from.

Figures 2, 3, & 4 show scatter plots of the percent of devices in Good State vs. Time. The graphs depict the time it took for the simulations with the different number of devices to reach the various percentages of good state. The names shown for each series, in the legends to the graphs represent the number of devices present in the simulation. The good state requirement for all the simulations was 65%.

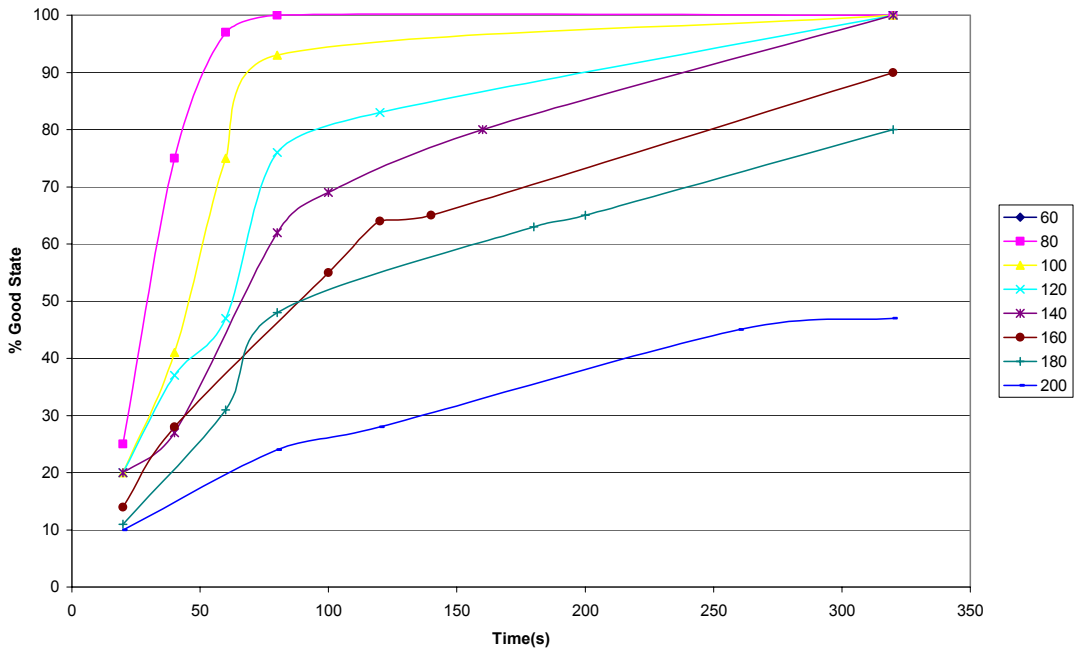

Figure 3: Good State vs. Time (Random)

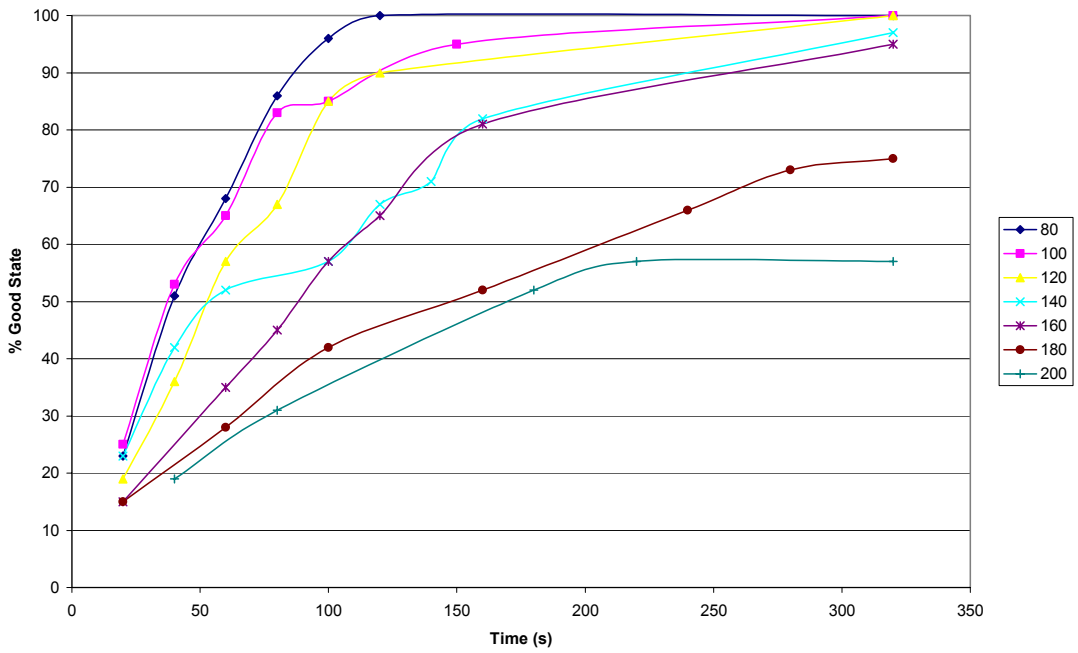

Figure 4: Good State vs. Time (Random w/ History)

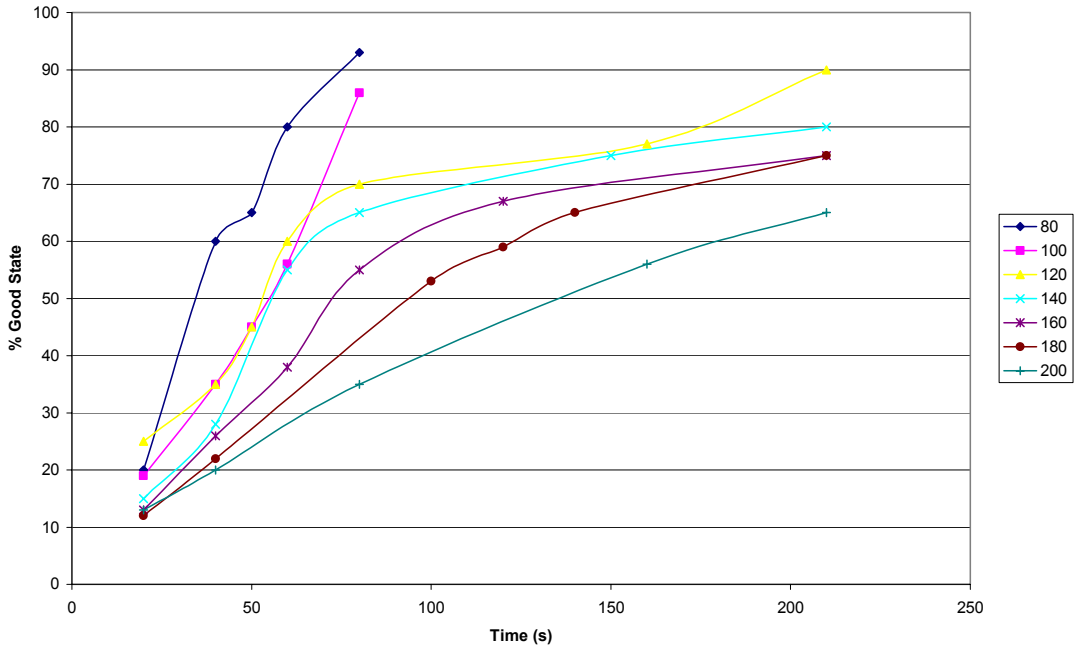

Figure 5: Good State vs. Time (Coordinated)

It is clear from the graphs that the coordinated protocol performed the best in terms of the time taken to reach the good state. For a good state of 65 % all the protocols

scaled linearly. The degree of linearity of the random and random with history protocols degraded with an increase in the number of nodes.

The figure 5 is a histogram of the average number of steps taken by each device, to reach a good state of 65%. The x-axis represents the number of devices in each simulation.

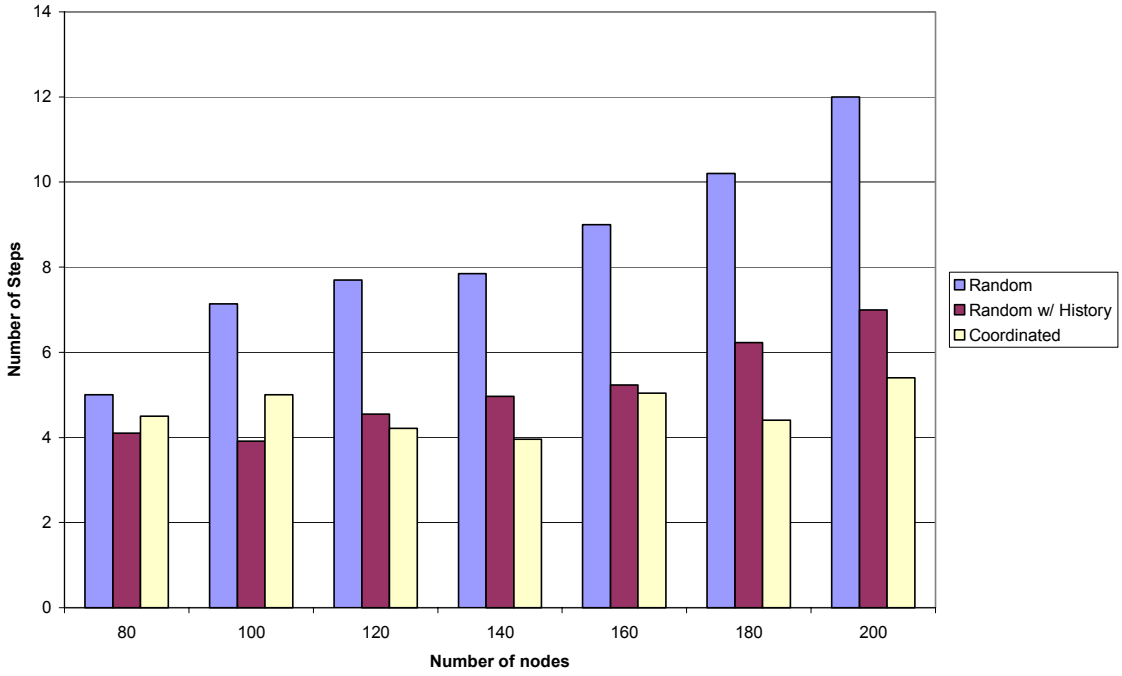

Figure 6: No. of steps taken to reach Good State vs No. of nodes

The histogram shows that the devices following the coordinated protocol require the least number of steps to reach the good state. The random with history protocol came in second in terms of this performance measure and the random protocol last.

Figure 6 is a histogram that shows the per device percentage of battery consumed for simulations with different number of nodes, following different protocols.

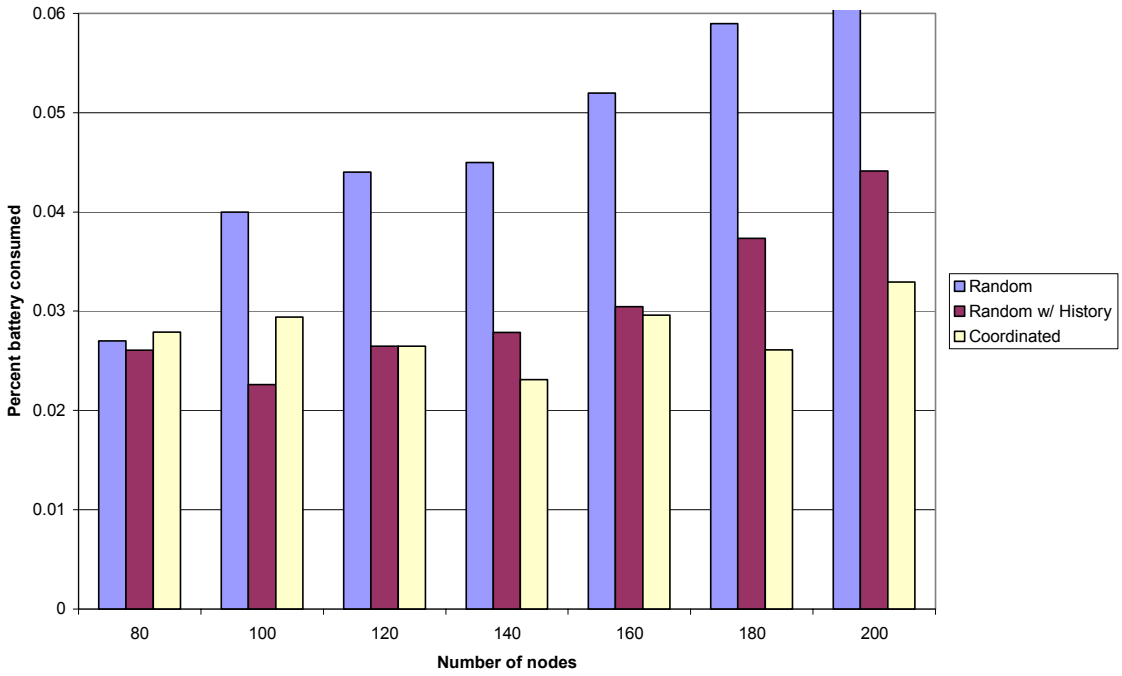

Figure 7: Battery consumption vs. No. of nodes

The coordinated protocol has the best figures in terms of the per unit percentage of battery consumed. The devices simulating the random with history protocol consumed slightly more energy compared to the coordinated protocol, whereas the random protocol consumed almost double the amount of energy. Also the per unit energy consumed by the coordinated protocol remained in the ball park region of 0.25% to 0.35 % whereas the values for the other protocols increased with an increase in the number of nodes in the system.

3.2.2 Corner Configuration

In this configuration all the devices were placed in one corner of the terrain. This initial configuration restricts the initial movement of the devices to 90 degrees and therefore dispersion takes longer.

Figures 7, 8, & 9 show scatter plots of the percent of devices in Good State vs. Time. The graphs depict the time it took for the simulations with the different number of devices to reach the various percentages of good state. The names shown for each series, in the legends to the graphs represent the number of devices present in the simulation.

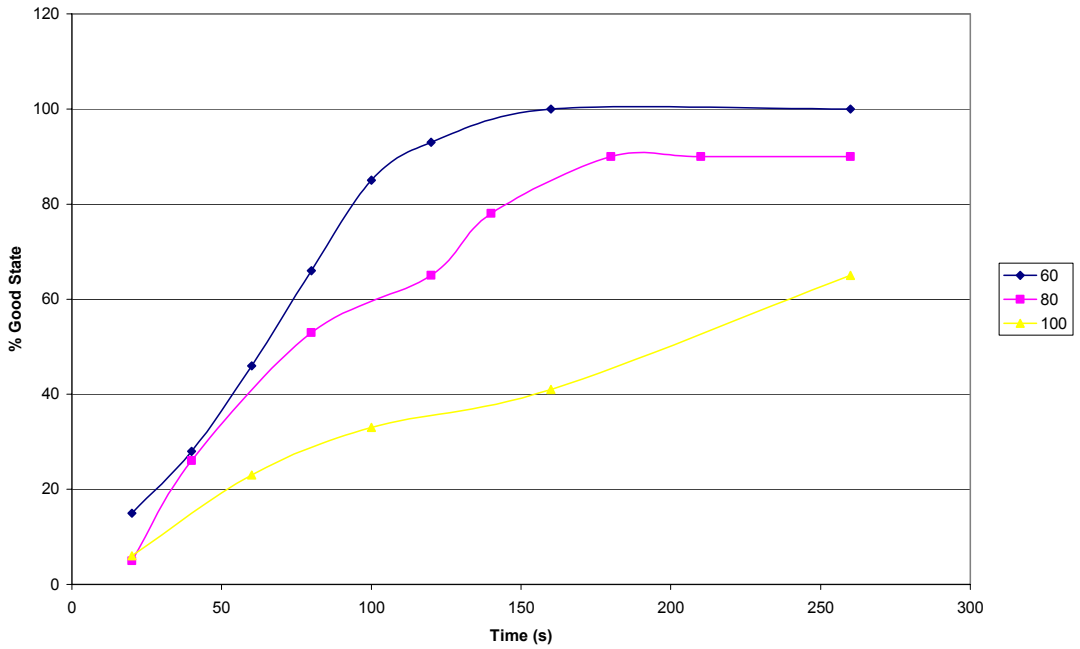

Figure 8: Good State vs. Time (Random)

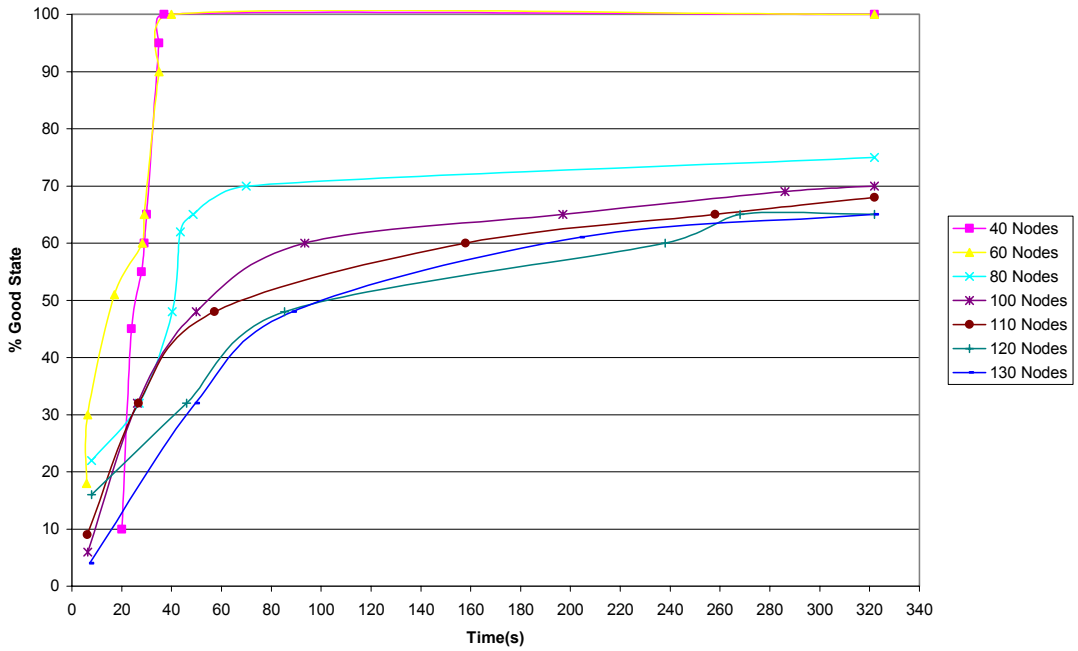

Figure 9: Good State vs. Time (Random w/ History)

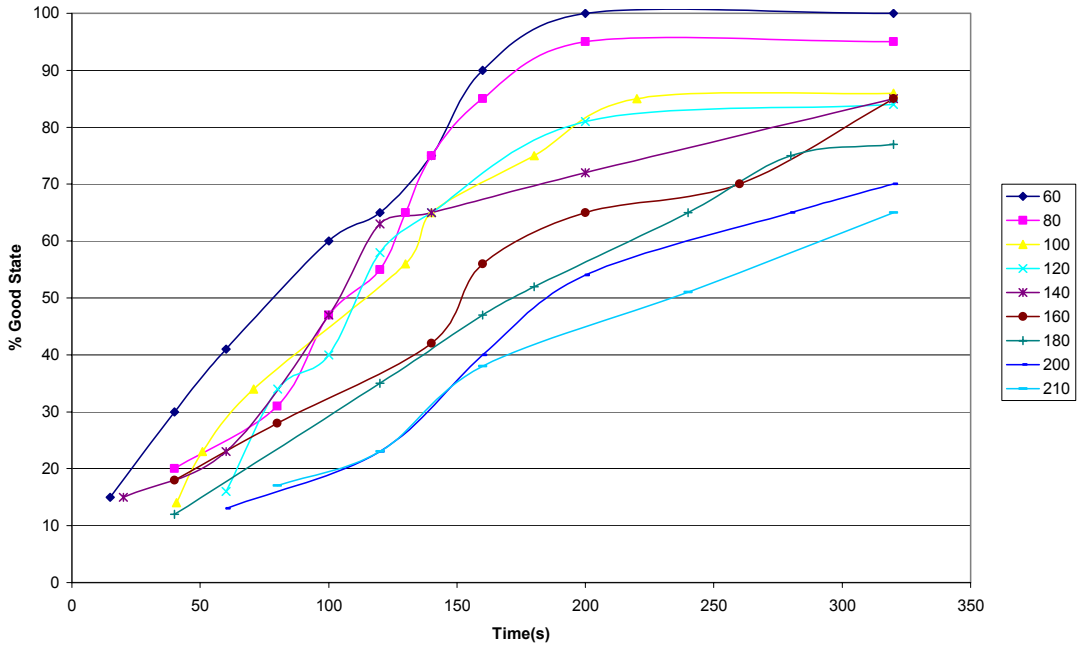

Figure 10: Good State vs. Time (Coordinated)

The results for the coordinated protocol are clearly the best, and for the 65% good state the behavior of the protocol is linear. The random protocol could only simulate up to a 100 nodes, and simulations with more than a 100 nodes never reached a good state of

65 %. The random with history protocol could simulate up to 130 nodes, and the performance quickly changed from a linear to one that is exponential in nature.

The figure 10 is a histogram of the average number of steps taken by each device, to reach a good state of 65%. The x-axis represents the number of devices in each simulation.

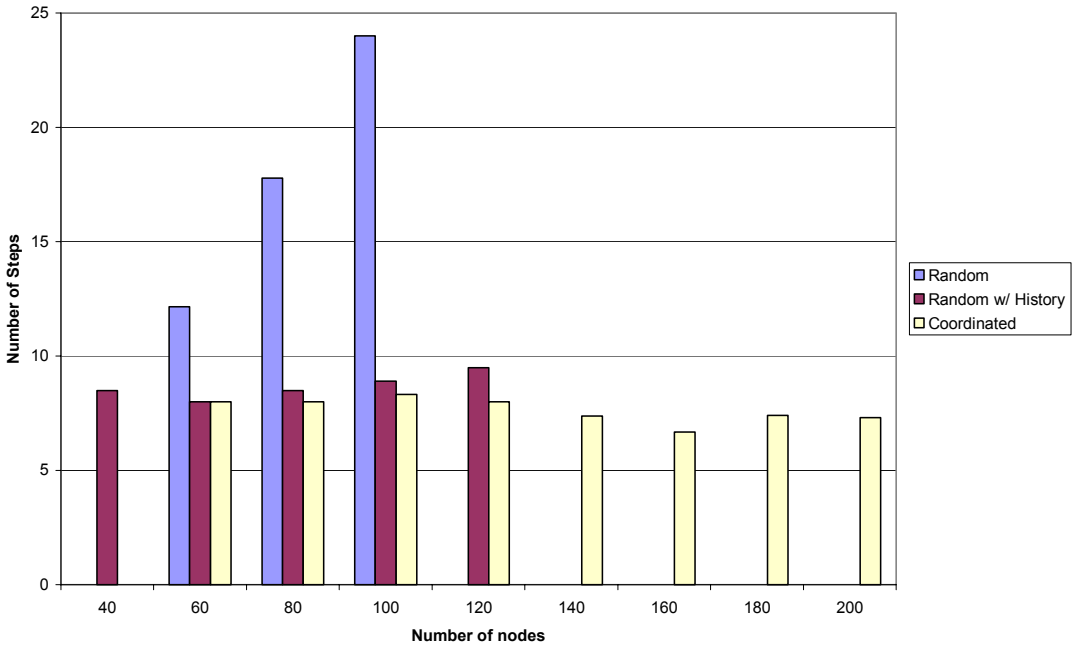

Figure 11: No. of steps taken to reach Good State vs. No. of nodes

The average number of steps taken by the coordinated protocol remained around the 7 – 7.5 mark. The random with history protocol also performed similarly but only up to a 120 nodes. The random protocol performed the worst. Not only did it require more number of steps to reach the 65% good state, but the values increased with an increase in the number of devices in the system.

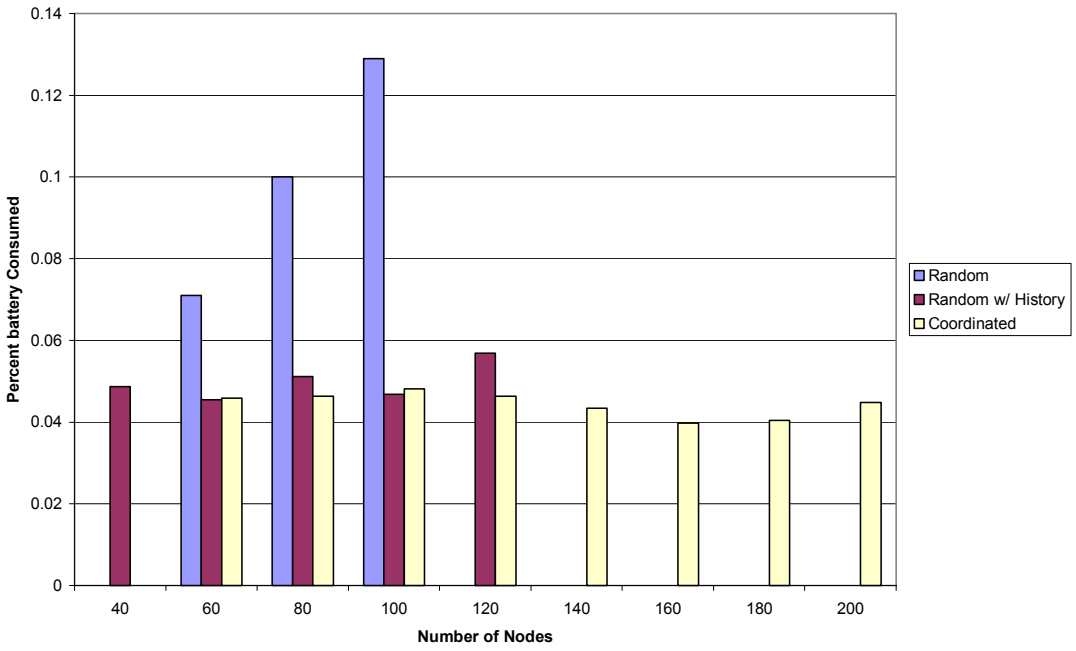

Figure 12: Battery consumption vs No. of nodes

The per unit percentage of battery consumed is the least for the coordinated protocol. The random with history protocol also performed similarly, but only up to 120 nodes. The random protocol has increasing values for the per unit percentage of battery consumed.

3.3 Mathematical Model

The analysis involved the calculation of the number of steps based on a circular arrangement. This is because for any given area a circular arrangement is the most efficient. If the initial placement of the devices is considered to be the center of concentric circles then for N devices N – 1 devices have to move with one device taking the center position. The devices arrange themselves around the circumference of these concentric circles. The radius of each circle is twice that of the previous one, with the initial circle having a radius(R) equal to the neighbor range. For example 50 meters was

the neighbor range for this project, and therefore a device would be considered a neighbor if it is within 50 meters of another device. Each concentric circle will have FLOOR (2∏R / neighbor range) number of devices arranged around it. This is because the devices have to be placed 50 meters apart from each other, so that they are in the dispersed state with 4 or fewer number of neighbors. Thus the first circle will have six devices around it; the second will have twelve and so on.

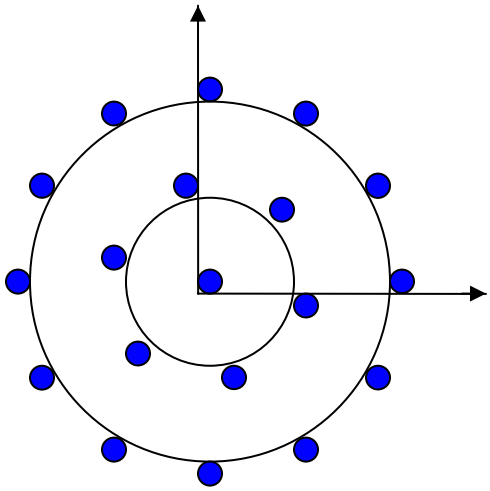

Figure 13: Circular arrangement

Therefore six devices have to move 1 step in order to reach the dispersed state, twelve have to move 2 steps, and the sequence continues. The number of steps moved will depend on whether the initial configuration is the center or the corner configuration. For the corner configuration the number of devices per circle will be FLOOR (∏R / (2*neighbor range)) and therefore the number steps required will increase. Therefore the first circle will have one device, and the second three, and so on.

Table 1 shows the optimal values for the center as well as the corner configurations. The values clearly exemplify the fact that the center configuration will require fewer number of steps compared to the corner configuration, to reach the dispersed state.

Number of devices	Center configuration	Corner Configuration
60	1.95	3.9
80	2.275	4.5
100	2.535	5.02
120	2.76	5.5
140	3.1	5.96
160	3.2	6.39
180	3.38	6.78
200	3.562	7.1

Table1: Optimal values for the number of steps required to reach a 65% good state

4.0 Conclusion

This chapter concludes the report by providing a summary, an interpretation, and recommendations for future work. The first section covers the important results in a succinct manner. The second section on the interpretation is the most important, as it analyzes the results, and puts them into perspective with respect to the optimal parameters. The final section discusses possible future work related to this topic.

4.1 Summary

The results of the three protocols varied with the initial configuration of the devices. The results for the center configuration were better than that of the corner configuration. The good state requirement for the simulations was 65%. The coordinated protocol had the best results for both, the center configuration as well as the corner configuration. Its values for the average number of steps taken and per unit percentage of battery consumed remained the same even as the size of the system was increased. The random with history protocol had similar values for the performance measures but only up to a certain number of nodes. The protocol did not scale. Same was the case with random protocol but its performance was even worse. The values for the average number of steps taken and per unit percentage of battery consumed increased as the system expanded.

4.2 Interpretation

The results clearly delineate the fact that the coordinated protocol was the best out of the three protocols that were tested. It took the least amount of time, energy, and also

had the lowest average number of steps. The random with history protocol was the second best and the random protocol had the worst performance. The results indicate that the more information each node has, the better the dispersion. For a good state requirement of 65% the coordinated protocol remained linear in its behavior with respect to time as the number of devices in the system was increased. The random and the random with history protocols started off as linear functions of time but became exponential as the number of devices was increased.

Figure 12 shows the above interpretation in a graphic form. The graph shows the linearity of the coordinated protocol as well as the exponential behavior of the random and random with history protocol.

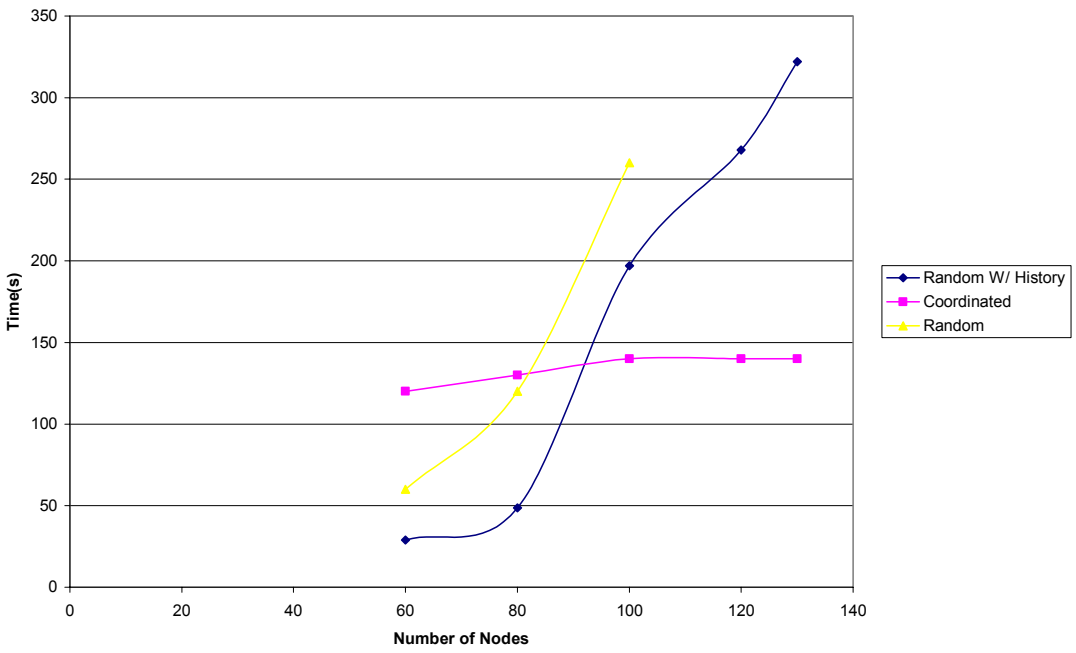

Figure 14 : Time take for 65% Good State vs. Number of nodes

The results for all the protocols were better for the center configuration than the corner configuration, and this indicates that the initial configuration of the devices within terrain does matter.

An interesting thing about the results for the corner configuration is that in case of fewer number of nodes the time required by the random and random with history protocols was much less compared to the coordinated protocol. One possible explanation could be the presence of fewer number of nodes. This because the coordinated protocol is dependent on the fact that various nodes will receive different step values and the choices made on basis of these received messages will result in better dispersion. Therefore the presence of fewer devices means that the number of different step values broadcasted will be fewer, and the dispersion may not take place in the fastest possible manner.

Another interesting thing to note is that average battery consumption per device for the coordinated protocol remained almost the same even as the system expanded. This was because as the number of nodes in the system increased, the proportion of nodes not moving also increased. Such nodes consumed very little energy, and thus the average amount of energy consumed by each device remained the same. The battery consumption per device for the random and the random with history protocolss increased with an increase in the number of devices being simulated.

Besides the relative comparisons of the protocols I used the number of steps taken on average by each device to reach the good state as a measure of how well a protocol performed. On comparing the simulation results with the optimal values (calculated in the previous chapter), we find that both the random and the random with history protocols are both, quite a bit off from the optimal performance. Consider a system of 180 nodes

with center configuration. It took 10.2 steps on average by each node using the random protocol, to reach the good state. Where as the optimal number of steps for this system are 3.38. The random protocol with history took 6.3 steps for the same configuration. Although this value is better than that of the random protocol, it is only the coordinated protocol that comes close to the optimal performance. With a value of 4.4 for the number of steps, it is only off by 1.02 steps from the optimal value.

Finally, the coordinated protocol is more stable. By this I mean that the variability in the performance of the coordinated protocol is marginal. On the other hand both the random and the random with history protocols have greater variability. On certain occasions their performance can be very poor whereas on others it could be even better than the coordinated protocol.

4.3 Recommendations

There is a lot that can be done to improve the results of this thesis. The first thing to do would be to improve the performance of the coordinated performance. This is because the performance of the coordinated protocol deteriorates with the increase in the percent good state. The performance is only linear up to the 65 % good state. So if 80% is the target good state then the current coordinated protocol will not be a viable solution. One possible way to improve it would be to have the protocol divide the system into small groups of devices with each group having a leader. The leader should be in charge of assigning the direction and the number of steps to each of its neighbors.

Another thing to further this project would be to implement the concept of a central base station within the disperse application. The base would send out intermittent

signals, which would be used to delineate the terrain for the swarm devices. For example, if a device is able receive the signal then it is within the terrain otherwise the device has moved out of the area of action. In this case the device should move till it enters the terrain again. Also it would be nice to get the graphical interface of GloMoSim running. It will be helpful to see how the protocol is working in real time, plus visual outputs always look better than numerical statistics. The next stage after this would be to convert the protocols into code for actual devices and run tests to see if the performance of the protocols matches the results of the simulations.

Bibliography

"Battery Specialist." Available: http://www.batteryspecialist.com/Merchant2/ /energizer.html#top

Cuvelier, Michael J. "Communication Aware Swarm Dispersion Primitives." Thesis. U of Virginia, 2002.

"Direct Drive DC Motors." Available: http://www.opticalparts.com/motors/direct.htm

Evans, Dave. "Programming the Swarm." U of Virginia. July 24,2000.

Homsy, George, Thomas F. Knight, and Radhika Nagpal. "Programming Paradigms for Amorphous Systems." 2000. ACM publications.

Jay, Martin. "About GloMoSim." Online. Internet. 7 Feb. 2001. Available: http://pcl.cs.ucla.edu/projects/glomosim/

Appendix A1: Simulation Results

Center Configuration

Random Protocol

Number of Devices	Time (s)	Energy (J)	Number of Steps
80	40	669	5
100	60	1260	7.14
120	80	1626	7.7
140	100	1950	7.85
160	140	2580	9
180	200	3300	10.2

Random with History Protocol

Number of Devices	Time(s)	Energy (J)	Number of Steps
80	60	642	4.1
100	60	696	3.91
120	80	978	4.55
140	120	1200	4.96
160	120	1500	5.23
180	240	2070	6.23

Coordinated Protocol

Number of Devices	Time(s)	Energy (J)	Number of Steps
80	50	672	4.5
100	65	900	5
120	70	978	4.21
140	80	996	3.96
160	120	1458	5.04
180	140	1446	4.4
200	210	2028	5.4

Corner Configuration

Random Protocol

Number of Devices	Time(s)	Energy (J)	Number of Steps
60	60	1328	12.16
80	120	2497	17.78
100	260	4000	24

Random with History Protocol

Number of Devices	Time(s)	Energy (J)	Number of Steps
60	29	840	8
80	48.6	1260	8.5
100	197	1440	9
110	258	1740	9
120	268	2100	9.1
130	322	2400	9.4

Coordinated Protocol

Number of Devices	Time(s)	Energy (J)	Number of Steps
60	120	846	8
80	130	1140	8
100	140	1482	8.33
120	140	1710	8
140	140	2000	7.8
160	200	2100	8
180	240	2400	7.4
200	280	2760	7.4
210	320	2844	7.3