A Communications Modeling System for Swarm-Based Sensors

Brian A. Kadrovach

AFIT/DS/ENG/03-03

A Communications Modeling System for Swarm-based Sensors

DISSERTATION

Brian A. Kadrovach
Major, USAF

AFIT/DS/ENG/03-03

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/DS/ENG/03-03

A Communications Modeling System for Swarm-based Sensors

DISSERTATION

Presented to the Faculty of the Graduate School of Engineering and Management

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy in Electrical Engineering

Brian A. Kadrovach, B.S.E.E., M.S.E.E.

Major, USAF

September, 2003

AFIT/DS/ENG/03-03

A Communications Modeling System for Swarm-based Sensor Networks

DISSERTATION

Brian A. Kadrovach, B.S.E.E., M.S.E.E.

Major, USAF

Approved:

Gary B. Lamont	5 Sept '03
Dr. Gary B. Lamont Dissertation Advisor	Date
[illegible]	3 Sep 2003
Dr. Richard A. Raines Committee Member	Date
Mark E. Oxley	3 Sept 2003
Dr. Mark E. Oxley Committee Member	Date
[illegible]	3 Sept 2003
Dr. Robert L. Ewing Committee Member	Date
[illegible]	3 Sept '03
Dr. Curtis H. Spenny Dean's Representative	Date

Accepted:

[illegible]

Robert A. Calico, Jr.
Dean, Graduate School of Engineering and Management

Acknowledgements

The individuals–students and faculty alike–that contributed to the success of this research are far too numerous to mention by name. To all of you I express my sincere appreciation for your patience, correction, and general good nature while I foundered about with homework, projects, and studies!

To my advisor, Doctor Lamont, I must say that your guidance and patient prodding are appreciated more than you may know. I thank you for the hours and hours of reading and reviewing my work knowing that mine was not the only task to which you so diligently applied yourself. I could not have done this without your expert helping hand.

I would like to thank my mother and step-father for being patient with me. You provided a solid foundation from which I could stretch to the heavens! I want to thank my father for giving me the dream of what I might be. Without your support and encouragement I would never have made the "first" step!

To my two favorite sons in all the world there is not a day that goes by that I do not thank the heavens for the joys you have both brought me during the difficult past three years. Thank you!

Finally, I would like to express my deepest and profound appreciation to the one person that has been my strongest supporter, my wife. Your patient endurance of what can only be described as the "AFIT experience" was angelic! I thank you from the bottom of my heart. Let's go for a leisurely walk!

I must also express my thanks to God for the provision of the entire support system I described above. Without His hand of Providence, none of this would have been possible!

So we fix our eyes not on what is seen,
but on what is unseen.
For what is seen is temporary,
but what is unseen is eternal.
II Corinthians 4:18

Brian A. Kadrovach

Table of Contents

Page

Page

Page

List of Figures

Figure Page

Figure Page

Figure Page

Figure Page

List of Tables

List of Abbreviations

List of Symbols

Symbol		Page
d_{min}	Ideal separation distance	4-3
S	The set of mobile particles	4-4
N	The population size (mobile particles)	4-4
p_i	The i^{th} particle in S	4-4
S_i	The set of particles in the neighborhood of p_i	4-4
N_i	The number of particles in the neighborhood of p_i, $\lvert S_i \rvert$	4-4
$\boldsymbol{p}_i$	Position of particle p_i	4-7
$\boldsymbol{v}_{update}$	Particle update vector	4-7
$\boldsymbol{v}_{attract}$	Particle attraction vector	4-7
$\boldsymbol{v}_{align}$	Particle alignment vector	4-7
$\boldsymbol{v}_{dir}$	Particle direction vector	4-8
$\boldsymbol{v}'_{dir}$	New particle direction vector	4-8
$\boldsymbol{p}'_i$	New position for particle p_i	4-8
w_d	Distance weight	4-8
w_{periph}	Peripheral vision weight	4-8
d	Distance between two particles	4-9
R'_2	Boundary and Waypoint region of *zero* influence	4-9
R_4	Particle region of *very far* influence	4-9
R'_1	Boundary and Waypoint region of non-zero influence	4-9
d_3	Radius of regions R'_1 and R_3	4-9
d_2	Radius of region R_2	4-9
c_{zone}	Comfort zone factor	4-9
R_2	Particle region of zero influence	4-9
R_3	Particle region of *far* influence	4-10
$C_{boundary}$	Boundary weight factor	4-11

AFIT/DS/ENG/03-03

Abstract

Today's information age has exploded the amount of data available to decision makers at all levels of the control hierarchy. The miniaturization and proliferation of sensor technology has enabled extensive detection and monitoring, and advances in computational capabilities have provided for embedded data analysis and the generation of information from raw data. Additionally, with the miniaturization of mechanical systems, it is possible to provide platforms for sensor suites that are capable of mobility and limited autonomy. Swarming, or bio-emergent behavior, provides a robust, scalable mechanism for organizing large numbers of mobile sensor platforms. However, the mobility dynamics of swarm systems present additional challenges.

This research develops a novel ad hoc data network communications modeling methodology for swarm-based sensor systems that provides a process for evaluating performance of communications protocols with respect to swarm dynamics. A new parameter-based swarm simulation system based on innovative vision models is developed and used to investigate and characterize swarm behavior. The process allows for communications protocol evaluations in the context of dynamic swarm behaviors.

Three network communications protocols are presented for swarm-based sensor networks and a performance comparison is made. The three protocols—Directed Diffusion, Geographical Routing Protocol, and Flooding Protocol—are compared. Results indicate, for the degree of mobility investigated, that the Directed Diffusion protocol slightly outperforms the Geographical Routing Protocol system. The swarm network modeling process developed provides a new methodology for rigorous and repeatable investigation of network communications systems with respect to the complex dynamics of swarm-based sensor networks.

A Communications Modeling System

for Swarm-based Sensors

I. Introduction and Overview

The following excerpt is taken from a description of *Prey*, a current fictional novel by Michael Crichton (31) about a particle swarm:

> *In the Nevada desert, an experiment has gone horribly wrong. A cloud of nanoparticles–microrobots–has escaped from the laboratory. This cloud is self-sustaining and self-reproducing. It is intelligent and learns from experience. For all practical purposes, it is alive.*
>
> *It has been programmed as a predator. It is evolving swiftly, becoming more deadly with each passing hour.*
>
> *Every attempt to destroy it has failed.*
>
> *And we are the prey.*

It has long been noted that what begins as fiction often finds itself in reality. This is the power of human imagination. Crichton's work describes a dire picture of technology gone awry. However, the reality of swarming technology is that it possesses significant potential for both commercial and military applications such as search and rescue operations or surveillance activities. The advances in computing and manufacturing technology have made such systems possible. In order to harness this potential for Air Force applications, a method of analyzing and exploring swarm systems–specifically swarm based sensor applications–must be developed.

Information processing (processing, storing, visualizing, disseminating, etc.) is one of today's leading engineering challenges. Military use of information is no exception. There is an immense amount of information available to today's military decision-makers from numerous sources including existing command and control systems, reconnaissance data, satellite data, unit capability data, and real-time battlefield conditions (124). Traditional communications networks are bandwidth limited especially in wireless networks (119). The problem: there are large amounts of data that need to be transferred over limited communications resources. These conflicting characteristics create a need for an efficient and effective communications system for large numbers of sensors in a communications resource limited architecture.

1.1 Research Definition

The amount of information that must be transferred in data dissemination networks is growing at an increasing rate. This is primarily due to the increase in sensor capabilities. Traditionally, sensor data is processed off-line and the results are propagated through the network because of the network bandwidth limitations. This creates a significant latency between event occurrence and event detection/notification. Currently, with the increases in bandwidth, the desire is for higher fidelity information along with the traditional analysis results. These sensors produce such data streams as streaming video, still images, and other image-type data. An example of this is synthetic aperture radar images. These images are on the order of 10MBytes (71). The traditional analysis results consist of files on the order of only several kilobytes. The high-fidelity types of information are found on many networks including military communications systems.

Additionally, great strides are being made in miniaturization of electronic and electro-mechanical systems (12, 67, 104, 126). These devices, equipped with wireless communications elements, multiple types of sensors, and varying degrees of mobility can provide sensor data in numerous environments including those unsuitable for traditional sensor systems. A large number of these devices, on the order of 100's or even 1000's, could work together like a swarm of insects or a flock of birds to provide high fidelity information on a near real-time basis.

Swarm or *emergent behavior* systems (14, 69, 99) such as swarms of insects or flocks of birds present a unique implementation method for a sensor system with a large number of individual sensors. Swarm behavior, like that seen in bee swarms or flocks of birds provides a stable organization of sensor platforms that is flexible, able to adjust rapidly to changing environmental conditions. These systems also provide graceful degradation when individual sensors fail.

The communications system for wireless ad hoc sensor networks must provide for the effective and efficient transfer of large amounts of data in a highly dynamic network environment. *Effectiveness*, in this context, is defined as the ability to successfully send data to the intended destination while *efficiency* is defined as higher throughput and lower latency. The communications issues involve optimal use of limited bandwidth resources in a dynamic, ad hoc network topology. An ad hoc network (30) is a collection of wireless mobile nodes dynamically forming a temporary network without the use of any existing network infrastructure or centralized administration (38). In such a network, each mobile node not only acts as a host but also as a router, forwarding data for other mobile nodes in the network that may not be within direct wireless transmission range of each other. In this manner, communications are enabled between members of the swarm as well as external hosts.

Potential communications systems that can support this research are based on the Joint Battlespace Infosphere (JBI) (124) or the Network Embedded Software Technology (NEST) (34). The JBI is an information management system for military commanders and their subordinates. Its purpose is to provide *appropriate* information (in terms of scope, detail, security, etc.) to every echelon of the command structure. The appropriateness of the information is determined by the needs of the system users. The system users access the JBI information via JBI clients. The purpose of the JBI is provide an information dissemination system that manages security and authorization issues while still being responsive to user needs. The user needs include timeliness and access.

The NEST program (34) is a research effort sponsored by DARPA to address the technical challenges for resource-constrained networks of embedded nodes. The DARPA Broad Agency Announcement for NEST states, "Embedded information processing is fast becoming the primary source for superiority in weapons systems. The current trend is toward "information rich" nodes with little separation between physical processes such as sensing and actuation and computational processes such as monitoring, diagnostics, and overall closed loop system control (34)." The NEST architecture is envisioned to provide fault-tolerant, self-stabilizing protocols for data exchange, synchronization, and replication in large scale (100 to 100,000 node) distributed, real-time systems.

Figure 1.1 depicts a high-level notional view of the use of a swarm based sensor system for some form of a search activity. The JBI is used to identify the intended track

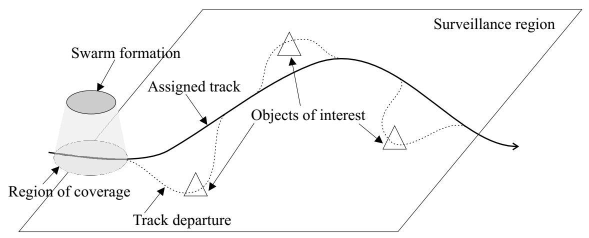


Figure 1.1 Notional Sensor Swarm System

through a particular surveillance region and assign the mission to the swarm. In the sense of swarm behavior, the assigned track is a desired global route through the region with some measure of leeway given to the swarm to investigate objects of interest. The route is termed *global* since the path is intended for the swarm as a whole. The swarm identifies objects of interest and redirects the route to provide a “closer look” (i.e. higher resolution, more sensors, etc.). The *redirection* results from the cooperative behavior of the swarm through an entirely distributed process–much like the way a flock of birds finds food sources(69). The closer look results in higher fidelity sensor data being generated by the swarm-based sensors. This data is routed back in near real-time to JBI clients in order to provide for dynamic mission planning.

The example shown in Figure 1.1 presents the swarm as a two dimensional structure. However, swarm formations might be organized into a three dimensional structure. This is more realistic for airborne swarm systems and is not an issue for ground-based systems. The use of a three dimensional formation for ground surveillance applications results in an increased redundancy and a reduced efficiency. This trade-off is an important aspect of swarm systems. Efficiency can be sacrificed in order to increase redundancy and thereby improve such things as reliability and survivability resulting in an overall improvement of effectiveness[1].

Self-organizing, distributed sensors (67, 103) and autonomous control systems (25, 89, 118) are areas of intense research. Developers at Sandia National Laboratories have

[1]Reliability and survivability are obtained at the *global* level which improves the likelihood of *mission* success–individual swarm members are not more or less likely to have increased reliability or survivability.

developed a technique using swarming techniques to improve searches for avalanche victims (103). Furthermore, the authors in (67) developed miniature sensor systems that have a passive communications capability. Such systems could be used in a wide range of applications including exploration of other planets, search for land mines, reporting on traffic bottlenecks, or as an aid to weather forecasting. These sensor particles can self-organize and report data from multiple sensors.

Swarm-based sensor systems are of interest to many activities where surveillance is crucial such as potential high-interest terrorist targets or even military battlefield information gathering. The JBI, for example, provides access for external users to potential swarm-based sensor systems (55) and the NEST research effort emphasizes the need for a robust communications system.

1.2 Sponsoring Organization

To this point, a case is made for the use of swarm-based sensors as well as how they can be integrated into Air Force applications. The Embedded Information Systems Engineering branch of the Information Directorate, Air Force Research Laboratory (AFRL), which is sponsoring this research, is advancing the state of the art for various embedded sensor systems. Their mission is to

> *...conduct research, develop, and demonstrate embedded information systems technologies and associated processes required to affordably engineer current and next-generation weapon and information systems capable of ensuring air and space superiority. Also, Embedded Information Systems Engineering develops adaptive/reconfigurable information systems capable of dynamically adapting to changes in mission or new threats and reconfiguring to perform different functions, or to enhance system fault-tolerance in support of the warfighter* (4).

The research effort presented in this document directly supports the mission for adaptive and reconfigurable information systems. Additionally, this effort supports the mission of the AFRL Control Systems Development and Application branch which is responsible for, among other things, the development of fault tolerant control system architectures and control automation (5).

1.3 Research Goal and Objectives

The goal of this research investigation is centered on developing a swarm-based sensor communications modeling process. Toward that end, a simulation-based swarm behavior system is developed. A swarm simulation system enables investigation of many types of swarm instantiations without the need for a physical system. A physical system is limited by cost in terms of resources and time. While technology is advancing in potential support of swarm-based sensor networks, there are currently no systems that can be used for direct measurement without extensive time and financial investments. Further, because of the complexity of network systems in general and swarm-based networks in particular, simulation is the preferred choice for initial system development and investigation. To be useful, the swarm simulation should be scalable with respect to behavior. This means that swarm behavior is consistent with respect to input parameters across a wide spectrum of swarm population sizes. Additionally, behavior should be quantitatively identified through an objective evaluation methodology. This provides for a formal behavior identification mechanism. Further, the communications model should seamlessly integrate the swarm behavior. Simulation models can provide significant insight into system function and therefore be used to evaluate potential implementation issues.

There are three major objectives of this work. The first is development of a swarm model, with extensive flexibility in manifested behaviors, to be used as a foundation for subsequent network communications analysis. Second, an analysis of swarm behavior and development of a classification methodology is made. Finally, network development and analysis is made in the context of swarm behavior. These three divisions provided the embodiment of this work: swarm model development, behavior classification, and network development and analysis.

1.3.1 Swarm Model Development. In order to accurately investigate network communications issues for a swarm-based sensor network system, a model of swarming behavior manifesting *realistic* member interactions is needed. Model development comprises the first part of this research effort with the specific objectives provided in Table 1.1. The model is developed so that behavior, as it relates to the configuration parameters, can

Table 1.1 Swarm Model

Objective: *Swarm Model Development*
– Investigate state-of-the-art swarming systems/applications
– Implement/improve swarm model
– Investigate swarm scalability issues

be investigated. Additionally, new characterizations (in the form of theorems and mathematical statements) of swarm formations with respect to particle member abilities (speed, maneuverability, etc.) are addressed.

1.3.2 Swarm Behavior Analysis. Behavior analysis is used to categorize various types of swarms. A relationship between swarm configuration parameters and the associated behaviors is generated based on a suite of behavior identification measures as summarized in Table 1.2. The proposed measures (see Chapter VI) provide a new set of

Table 1.2 Swarm Behavior Analysis

Objective: *Swarm Behavior Analysis*
– Develop/analyze behavior measures methodology
– Develop behavior classification methodology

tools for investigating the effect of configuration parameter changes on the swarm behavior. This provides a mechanism for quantifying swarm behavior and relating that behavior to the inter-particle dynamics that affect a network used to communicate sensor data throughout the swarm. The potential exists to apply these measures to any swarm system (modeled or physically instantiated) in order to characterize *global* swarm behavior.

1.3.3 Swarm Network Analysis. The literature reveals no swarm based network system. This research effort develops a novel network simulation methodology based directly on swarm movement patterns. The specific objectives are listed in Table 1.3. Significant work is already being done in the area of sensor networks and wireless, ad hoc network communications protocols (29, 58, 82, 115, 101), however, no research has directly linked swarming movement to network dynamics. This research develops that link and incorporates a scalable swarm movement model.

Table 1.3 Swarm Network Analysis

Objective: *Swarm Network Analysis*
– Develop simulation methodology for swarm based sensor network
– Develop comparison protocols
– Develop network evaluation methodology
– Analyze protocol performance in sensor swarm network
– Develop quantitative link between swarm behavior and network characteristics

There are several challenges that make this research difficult. First, there is a need to use realistic movement patterns for mobile sensor networks. One possible option is to generate movement patterns from observed actions of various swarming creatures in nature. While this has been done for simplistic systems (16, 91), it is too difficult to generate movement patterns from such swarms as a flock of birds or a swarm of insects from observation-based data. A realistic swarm simulator can be used to generate movement patterns for any number of particles for any length of time. However, this reveals another challenge. A graphics-based simulator can be used to study swarm behavior within the simulation but it relies on the user to determine the type of behavior–with respect to dynamics–that is generated. This hampers the ability to automate the simulation process. This leads to yet another challenge. No methodology exists that quantitatively describes swarming behavior. It is important to be able to remove the human-in-the-loop requirement for *autonomous* operation of the swarm. Further, any methodology that is used must be distributed in nature in order to be applicable to a swarming system. Finally, no mechanism exists to seamlessly integrate swarm movements into a network simulation system.

1.4 Research Approach and Scope

A methodical approach is used in satisfying the previously defined goal and objectives. A careful search of the background and state-of-the-art in swarm research and ad hoc mobile networking provides the foundation for development of the swarm simulation tool and network communication protocols.

The control aspects of swarms are not considered in this work–a control mechanism that implements swarm requirements is assumed. The focus is on developing a realistic swarm model–one that exhibits the behaviors of different types of swarms in nature–and using it as a testbed for evaluating swarm based networking protocols. The swarm is assumed to be a homogeneous collection of *platforms*, possessing a level of mobility and computational, communications, and sensory capabilities.

The typical communications network measures of throughput, latency, and delivery effectiveness (15, 119) are proposed for evaluating the performance of the sensor swarm network system. However, the input parameters are slightly more complex than simple bandwidth and overhead. Swarm platform capabilities (e.g. speed, maneuverability, and weight to name a few) determine the sensor network topology and the resulting connectivity among the swarm members. A complete list of swarm capabilities, parameters, and their effects is given in Section 4.1.

Data exchange between the external users and the swarm network system should conform to the JBI structured common representation (124). It is assumed that a standardized data exchange protocol such as the Extended Markup Language (XML) (45) and Sun Microsystem's JINI technology (117) play a significant role in the protocol development effort. These exchange protocols are widely used in numerous database and information dissemination applications (41, 90).

1.5 Document Organization

This dissertation is organized into nine chapters and several appendices. Chapter II provides background information that is necessary to understand the problem domain and the tools required to develop a solution in the application domain. Chapter II summarizes the JBI and gives pertinent information on particle swarm optimization, swarm modelling, wireless networks, and wavelet transform and data compression techniques. Chapter III provides a detailed discussion of contemporary work in the areas of sensor platforms, ad hoc sensor networks, and image fusion research. Chapter IV details the swarm model and provides a discussion of swarm configuration parameters and their effect on swarm behavior. Chapter V describes the communications protocols developed and compared in the swarm network. Chapter VI presents a suite of swarm and swarm network evaluation measures which are then used in Chapter VII to evaluate the swarm based sensor network system. Chapter VIII summarizes the results of this research and proposes several areas for potential future work.

Seven appendices are included to provide background and supporting data for the main document. The appendices provide background on wireless networking, communications simulator performance issues, as well as a summary of the pertinent swarm algorithm equations. Additionally, an appendix is included that describes function and use of both the graphics and command-line simulators developed as part of this effort. Two appendices are used to detail noise margin and connectivity analyses. Finally, an appendix detailing the various supporting programs and analysis tools is included as an aid to future work.

II. Foundations for Sensor Swarm Networks

This chapter presents an overview of the various research areas that constitute the foundation of this investigation. The details presented in this chapter establish the context in which this research is placed. Application areas for swarm vehicles and sensor networks are described. The focus for swarm vehicles is on airborne systems, specifically on Uninhabited Aerial Vehicles (UAV). The discussion of sensor networks is based on the Joint Battlespace Infosphere (JBI) (124) and the Network Embedded Software Technology (NEST) (34) efforts. Also included is an introductory discussion of particle swarm optimization. Finally, a description of ad hoc wireless networking and associated design and implementation challenges are provided.

2.1 Uninhabited Aerial Vehicles

Uninhabited vehicles (UAVs) are a novel addition to modern warfare. They were used extensively in Europe to support operations in Bosnia (123), and, more recently, in the second Gulf War. However, their development has a rich history beginning with Charles Perley's Unmanned Aerial Bomber (70). Developed in 1863, Perley's invention used a hot air balloon and basket to carry explosives. A timer mechanism was used to release the explosives over the desired target.

The first documented use of unmanned airborne assets for reconnaissance activities took place in 1898 during the Spanish-American war (70). Corporal William Eddy used a kite to carry aloft a camera that was then used to take pictures of enemy positions.

Numerous other examples of the use of UAVs throughout history exist up to and including the Israeli use of UAVs in the 1980s (70). The Israelis successfully used a fleet of UAVs called Scouts during the Bekaa Valley conflict between Israel, Lebanon, and Syria. The Scout, shown in Figure 2.1, is a lightweight fixed-wing aircraft with a fiberglass body, and its main missions were visible and infra-red video surveillance systems (59).

The Israelis continued to hone their reconnaissance capabilities through the development of the Pioneer, shown in Figure 2.2. Because of the Israeli successes, the United States military (Navy, Marines, and Army) acquired more than 20 of the Pioneer aircraft.

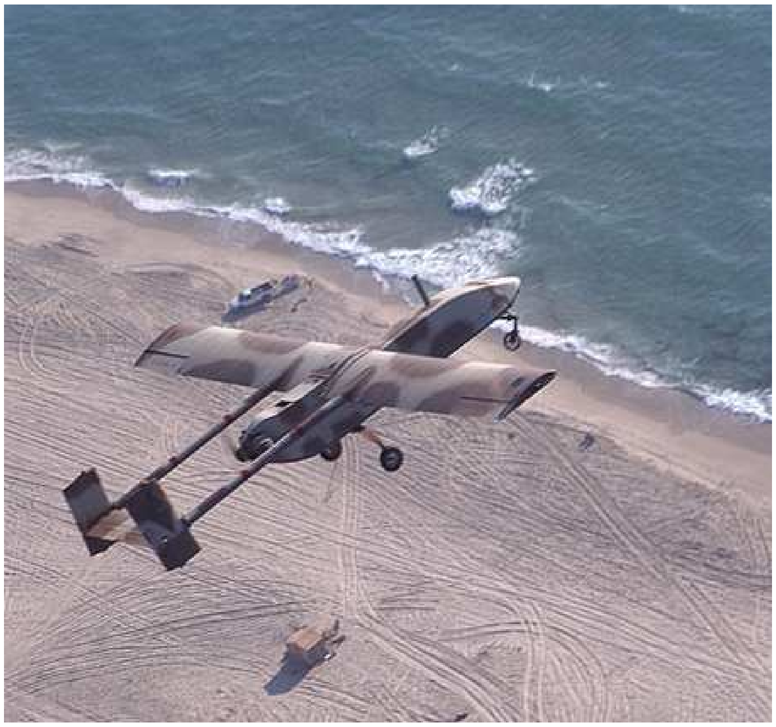


Figure 2.1 Israeli Air Force Scout (70)

These UAVs saw action in the first Gulf war and later, in Bosnia. The Pioneer has a proven record in working with the Air Force's Joint Surveillance Target Attack Radar System (6).

Today's inventory of U.S. UAVs consist of a list of successful applications of technology to military needs. These UAVs, shown in Figure 2.3, include the Pathfinder, Predator, and Global Hawk. Their functions range from environmental research missions to high-fidelity real-time reconnaissance missions to armed attack missions.

The Pathfinder UAV is designed for high altitude wind and weather research as well as providing high resolution digital images (70). The Pathfinder was recently involved in an experiment to test remote aerial imaging and analysis, wireless ethernet "bridge" communications, and commercial capabilities of UAV technologies (53).

Designed for long loiter times, the Predator UAV provides up to 40 hours of flight time (1) with surveillance capabilities that include high resolution color video, infra-red images, and synthetic aperture radar (SAR) (70). Though originally designed only for

Figure 2.2 Israeli Air Force Pioneer (70)

(a) Pathfinder (b) Predator (c) Global Hawk

Figure 2.3 Current U.S. UAV Assets

reconnaissance activities, several Predator UAVs have been equipped with antitank missiles and have successfully hit their targets (102).

The Global Hawk, while still an experimental UAV, has been used extensively in Afghanistan and, more recently, in Operation Enduring Freedom (70). The Global Hawk UAV is able to operate at 65,000 feet with a flight endurance of more than 30 hours. The UAV operates autonomously from the time it takes off to the time it lands (111).

The progression of UAV technology provides a sound foundation for development of airborne, swarm-based sensor networks. As platforms increase in capability (increased endurance and reliability, reduced size and cost), they become the enabling technology for instantiating a large swarm—on the order of 100s or even 1000s—of sensors.

2.2 Sensor Networks

The JBI and NEST efforts provide support for meeting the challenges of information management in dynamic communications applications. These efforts are potential application *umbrellas* for a swarm-based sensor network system.

2.2.1 Joint Battlespace Infosphere.

The JBI provides the answer to the challenges of these complex information management issues. In its December 2000 report on building the Joint Battlespace Infosphere, the USAF Scientific Advisory Board defines the JBI as follows (124):

> *The JBI is a combat information management system that provides individual users with the specific information required for their functional responsibilities during crisis or conflict. The JBI integrates data from a wide variety of sources, aggregates this information, and distributes the information in the appropriate form and level of detail to users at all echelons.*

The JBI provides the means for implementing the concepts put forth by *Joint Vision 2010* (62)–it enables getting the *right* information to the *right* user at the *right* time in the *right* format in the *right* language and at the *right* level of detail (124).

The term *JBI* has a dual meaning. When speaking of *the JBI* it refers to the definition presented. However, it is not envisioned that the JBI be an all encompassing information infrastructure but rather a system set up in response to a specific crisis or conflict. In this case *a JBI* is the set of components, links, policies, and users involved in a specific military operation. A simple view of a JBI is presented in Figure 2.4.

Figure 2.4 shows the key architectural components of the JBI. The *global grid*, consisting of heterogeneous network communications systems, provides connectivity for the JBI *platforms* and *servers*. The JBI servers provide support services that include security and management functions for the platforms.

The fundamental element of the JBI is the *information object*. Every piece of information is contained in an information object. A *client* is any computer system that interacts with the JBI. These clients *publish* or *subscribe to* information objects. A *user* accesses the JBI through the client. The *owner* of the JBI—the commander and the commander's information staff—sets policy for users and clients.

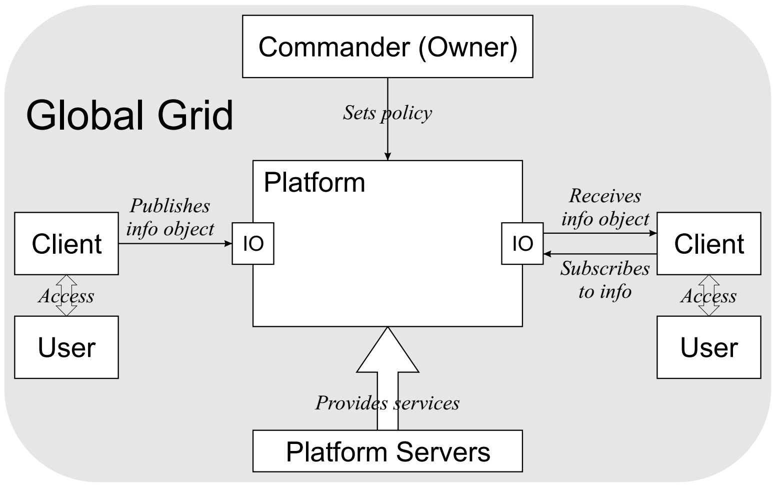


Figure 2.4 High Level View of JBI

2.2.2 Network Embedded Software Technology. An area of intense research focuses on embedding functionality within a network system. The Network Embedded Software Technology (NEST) effort—sponsored by the Defense Advanced Research Programs Agency (DARPA)—addresses the need for this research.

The goal of NEST is to develop a methodology that enables "fine-grain" fusion of physical and information processes. The design target is dependable, real-time, distributed, embedded applications that consist of 10^2 to 10^5 computational nodes. In these applications, the computing nodes are connected by a communications network and their operation is dynamically coordinated and reconfigured in response to changing environmental conditions (i.e., failure, external threats, etc.). To quote the DARPA solicitation announcement (34), potential applications include "MEMS-based control and health management of weapon platforms, coordinated operation and control of large groups of physical objects (weapons, munitions, vehicles), and smart structures."

Work by researchers in this area include development of an "Active Message" (125) communication system for networked sensors (54). This research is part of an overarching effort at the Wireless Embedded Systems laboratory at the University of California, Berkeley, which is focusing on development of a software/hardware platform for accelerated development of algorithms, services, and applications for the NEST community.

Additional research is being conducted in a joint effort between the Washington University, St. Louis and the Boeing Company (116). This work focuses on developing flexible and reusable *middleware* services for NEST architectures. The middleware provides support services that makes the underlying communications grid invisible to the user applications.

These technology development areas provide the much needed infrastructure to support swarm-based sensor applications. However, these developments are at a high level—much like TCP/IP protocol systems used in today's internet—and do not address the underlying physical network characteristics of a highly mobile and dynamic network system.

2.3 Particle Swarms

This section provides a discussion of particle swarms and includes a description of implementation techniques for particle swarms. Particle swarms are used in several application domains including optimization, data mining, and vehicle control. The research described here does not use swarms in the traditional sense of optimization search (69). However, for completeness, a brief discussion of particle swarm optimization issues is included in Appendix A.

The use of particle swarms in science and engineering research and applications attempts to reap the benefits of a process that nature has already evolved. Birds are able to fly in large flocks and maintain their positions with apparent ease. This same flock improves the foraging process by "distributing" the search for food or avoidance of danger so that the whole flock benefits. The advantage of swarms is a result of *collective* behavior. Partridge (91) states that collective behavior occurs when animals "move in unison, more as a single organism than a collection of individuals". Movement is dependent on the characteristics of the animal. For example, insects and birds can fly in three dimensions while sheep and ants are restricted to two dimensions. The environmental conditions that affect movement differ also. For example, the presence of prey or cold climates cause the swarm to behave in a completely different manner. Biologists propose several hypotheses for flocking behavior. It serves to reduce the risk of being eaten by a predator, provides mating efficiency, enables finding food easier and is a good environment for learning and reducing overall aggression(128).

Applying swarming principles to science and engineering problems falls into the category of *biomimetics*. The term *biomimetics* comes from the Greek "biomimesis," meaning to mimic life (114). The use of biomimetics by scientists and engineers is an attempt to take advantage of the efficiency inherent in natural systems. Applications include optimization (69), data mining (47), and control (25, 44, 92).

The use of swarming algorithms for data mining is an area of extreme interest. Data mining or, more specifically data clustering, is the process of grouping similar objects according to some set of characteristics. These characteristics include *distance*, *connectivity*, and *relative density* (47) and can be defined in the traditional spatial sense for locality applications or in a more abstract sense with respect to information. A good introduction to this research area can be found in (48).

Another area of ongoing research is the control of large swarms of vehicles (23, 43, 44, 51, 89, 92). Swarm control issues are important to this research because it establishes the physical network topology. This is critical to the development and analysis of the ad hoc network used for sensor data communications.

2.3.1 Swarm Behavior Rules. Reynolds (99) presents the classic swarm control theory in the description of his *boids* model. There are three basic control rules that govern movements of particles within the swarm. Each particle follows these rules based only on its *perception* of the environment. The control rules are presented in Table 2.1.

Table 2.1 Swarm Particle Behaviors

Behavior	Description
Separation	Avoid collisions with *nearby* particles
Alignment	Attempt to match velocity with *nearby* particles
Cohesion	Attempt to stay close to *nearby* particles

The emphasis on *nearby*–denoted as *neighborhood* in (99)–in Table 2.1 is important. Swarm behavior is based only on locally observable phenomena and therefore, particles can only *react* to swarm particles that are close in proximity. The definition of *nearby* is dependent on the application and is based on several parameters including speed, maneuverability, and size of swarm members. The rules described in Table 2.1 are illustrated in Figure 2.5. An animation that implements these rules in a simple Java application can be found at (98).

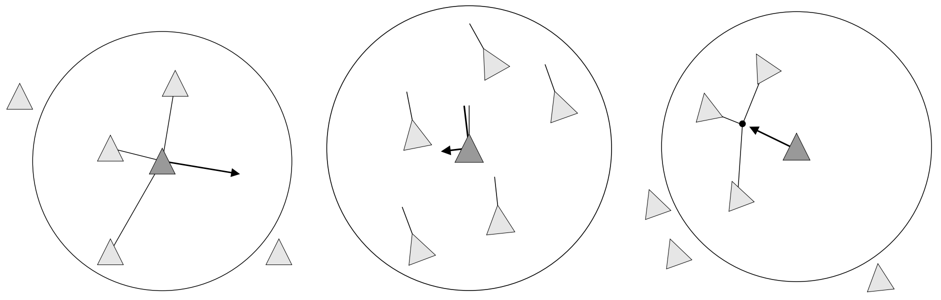

(a) Separation (b) Alignment (c) Cohesion

Figure 2.5 Swarm Behavior Rules

In Figure 2.5, *nearby* is defined to be the region within the circle centered on the swarm member of interest. In practice, *nearby* is often restricted to a section of the circle based on direction of travel and *peripheral vision.* This characterization of the neighborhood is problem domain specific. When discussing autonomous control of UAVs, an omnidirectional proximity sensor is often assumed (92) so that the neighborhood is indeed a circular region with the radius based on the sensor characteristics.

These swarm behavior rules, when implemented in a robust manner for swarm control, result in a stable swarm formation (whether flying, floating, rolling, etc.) where every member is at least some minimum distance from every other member and not any further than some maximum distance (as a result of separation and cohesion). The *alignment* behavior ensures that the swarm, as a formation, remains stable even in a dynamic environment. An example of swarm of 20 particles is shown in Figure 2.6. The particles are indicated by the large dots and their trajectories–over a period of 20 time units–are indicated by the solid lines. The edges between the particles indicate that the particles are not more than some maximum distance apart–in this case a distance of one unit. The model and associate parameters used to generate this sample swarm formation are described in detail in Chapter IV.

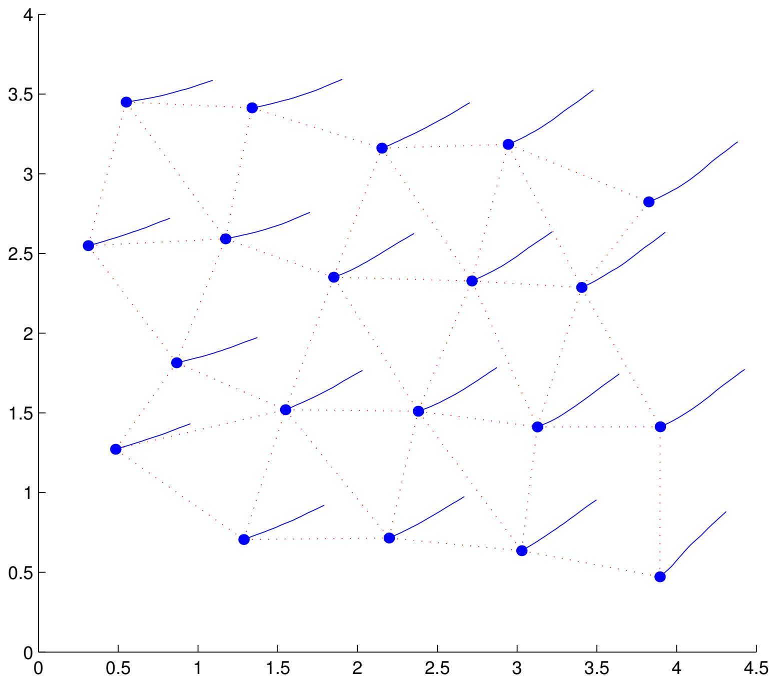


Figure 2.6 Swarm Formation

As Figure 2.6 demonstrates, the particles form a fairly well defined, regular formation based only on local interactions. It is this property of even dispersal over a particular region that makes swarm-based sensors advantageous for intelligence, surveillance, and reconnaissance (ISR) activities.

2.4 Ad Hoc Wireless Networks

This section provides an overview of networking and the challenges associated with wireless implementations. Wireless networking challenges include limited bandwidth and higher bit error rates–both of these qualities are orders of magnitude worse than that for traditional wired networks. A review of the network reference model developed by the International Standards Organization (ISO) and a brief discussion of potential communications implementations and issues are given in order to provide a basis for understanding the communications protocol systems developed as part of this research effort.

2.4.1 The Network Reference Model. The model developed by the ISO is called the Open Systems Interconnect (OSI) model. It standardizes the interconnection strategy between open systems (119). Figure 2.7 shows a notional network topology and the seven layers that make up the OSI reference model as given in (119). There could be any number of intermediate routers in the communications subnet indicated in Figure 2.7.

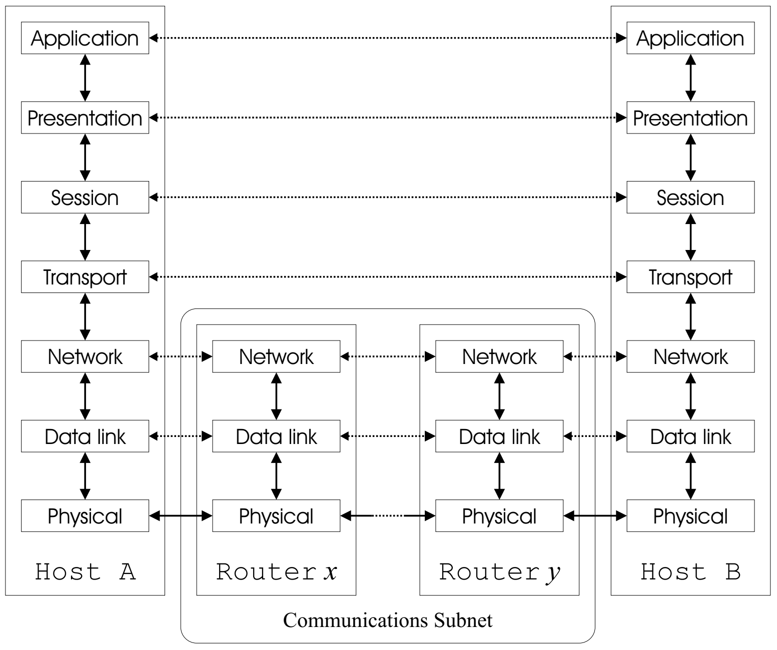


Figure 2.7 OSI Reference Model

All connections, with the exception of those at the physical layer, are virtual. This is indicated by the dotted lines in Figure 2.7. Information is transferred between hosts by establishing a virtual channel at each layer. For instance, at the application layer, an application on `Host A` requests a connection to an application on `Host B`. An example of this would be a file transfer protocol (FTP) client attempting to access an FTP server. The request from the client application on `Host A` for a connection with `Host B` is made to the application layer on `Host A` via the interface specified by that layer. The application layer on `Host A`, in some finite amount of time, responds with a confirmation that the connection with `Host B` is established. In this way, it appears to the application that there is a connection from `Host A` to `Host B` at the application layer. The same process described is used at each succeeding layer (again, with the exception of the physical layer).

The routers in Figure 2.7 represent the end-points of any number of intermediate routers between the two hosts. Routers are simpler in that the higher layers are unnecessary.

The use of the OSI layer model allows abstraction of the lower layers when considering various network design issues. In this research, models of the upper layers were developed while assumptions about and abstraction of the lower layers were used. Specifically, the bottom two layers (physical and data link layers) are modeled by a representative abstraction. For instance, the capabilities associated with the IEEE 802.11 wireless Ethernet standard (85) were used to provide the foundation for the network layer. There is a problem with explicitly implementing each layer of the OSI model. While the layer model provides standardization and compatibility, it requires significant overhead in order to manage the access between the layers. Wireless systems—which usually require low-power and resource constrained implementations—must trade off the flexibility of an explicitly implemented system with the resource savings that can be obtained from customizations of the communications system. A summary of several specific wireless implementations is given in Appendix B.

2.4.2 Simulators. There are several network modeling systems available to the developer. Two of those systems–OPNET and the Network Simulator, version 2–are used in this research. OPNET (87) is a commercial tool that, along with a modeling and simulation capability, provides an extensive graphical user interface for model development and system simulation. The Network Simulator (*ns-2*) (39) is a public domain simulator maintained and updated by an informal consortium consisting of a large user base and a small group of developers at the Information Sciences Institute of the University of Southern California (122).

Several recent developments have resulted in proliferation of large-scale parallel network simulators. The most notable of these include the GloMoSim (72, 131), the Dartmouth Simulator for Wireless Ad Hoc Networks (SWAN) (94), and a parallelized version of *ns-2* called *pdns* (100). These are parallel simulator applications intended to run simulations with 10,000 or more wireless nodes. This simulators are efficient for running large scale wireless simulations but possess insufficient mobility capabilities for modeling swarm-based sensor networks.

2.4.2.1 OPNET.

OPNET was initially developed at the Massachusetts Institute of Technology and commercially introduced in 1987 (88). OPNET uses a hierarchical approach to model objects. At the lowest level, objects are modelled as processes using finite state machines with behavior specified by C/C++ logic. At the next level, processes are grouped together and connected via streams to form node objects. Finally, nodes are grouped together to form networks. Nodes can be connected by physical or wireless links. Additionally, OPNET provides visualization tools to generate network systems using a drag-and-drop methodology as well as graphically display simulation results in several formats (88).

2.4.2.2 The Network Simulator, ns-2.

The network simulator *ns-2* is an object oriented, discrete-event simulator written in C++; it uses OTcl (39), an object-oriented form of Tcl (127), as a command and configuration interface to provide for rapid instantiation of new elements. The *ns-2* simulator supports complex objects decomposed into simpler components for greater flexibility and composability via the object-oriented version of Tcl. For advanced development, objects can be written in C++ with a command interface developed in OTcl. This provides for more efficient run-time execution of large, complex models (39).

For this research effort, the *ns-2* simulator is used. The reasons for this are two-fold. First, the Directed Diffusion protocol system (described in Section 3.2.2) is already implemented in *ns-2*. Second, the performance of *ns-2* for the wireless, swarm-based network system is better than OPNET. A performance analysis comparison conducted as a part of this research is given in Appendix C. The results of this analysis indicated that, while equally effective with respect to simulation fidelity, the *ns-2* simulator, compared to OPNET, is more efficient with respect to compute time for large, complex ad hoc networking simulations by a factor of 3, for 20 nodes, and a factor of 16 for 100 nodes.

2.4.3 Wireless Network Challenges.

The greatest challenge to wireless networks is the limited bandwidth and high bit error rate (BER) (97). The bandwidth resource is limited since most wireless implementations use RF links and the RF spectrum is limited. Also, there is a great deal of competition for the RF spectrum and wireless computer networks are only a small part of that competition. Traditional wired networks are virtually error free so that lost information—usually in the form of lost packets—is almost always due to network congestion. Therefore, network flow control protocols, such as that used in the internet Transport Control Protocol (TCP), use lost packet measures to control how fast host computers are allowed to transmit packets (119). In a wireless environment, lost packets are usually due to bit errors caused by the unreliable wireless link (110). This causes unnecessary reduction in packet transmission rates and severely inefficient use of the limited available bandwidth (18). The following discussion provides a qualitative analysis of the challenges associated with wireless networks using TCP.

As stated, wireless links suffer from a high BER, and the errors tend to be bursty in nature rather than uniformly dispersed. The result of bursty errors is that the bit errors in a bit stream tend to be concentrated (grouped together). These types of errors are usually due to local effects such as lightning. Figure 2.8 is a notional view of the time varying BER for a wireless link.

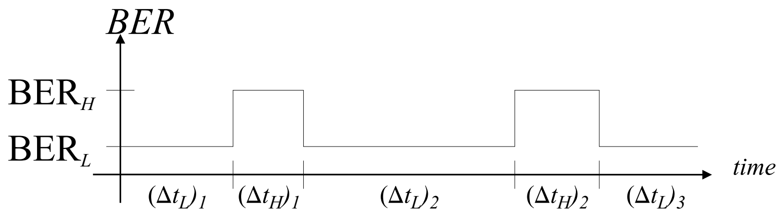


Figure 2.8 Bit Error Rate Model

Several assumptions were made in order to generate the shown plot. First, the overall average BER is constant (approximately 10^{-6}, as stated earlier). Also, it is assumed that the vast majority of bit errors are a result of burst errors and that the high and low BER values are statistically independent (but constant). The times that the two BER values are valid are assumed to be exponentially distributed (119) with means of $(\Delta t_L)_{ave}$ and

$(\Delta t_H)_{ave}$ for the low and high BER values respectively. The effective BER (BER_{eff}) of the wireless link can then be computed as shown in Equation 2.1.

$$BER_{eff} = \frac{(\Delta t_L)_{ave}}{(\Delta t_L)_{ave} + (\Delta t_H)_{ave}} \cdot BER_L + \frac{(\Delta t_H)_{ave}}{(\Delta t_L)_{ave} + (\Delta t_H)_{ave}} \cdot BER_H \tag{2.1}$$

If BER_L is assumed to be zero and BER_H is assumed to be non-zero, then the effect of burst errors on a wireless link can be demonstrated. Making the appropriate substitutions and algebraic manipulation, Equation 2.2 is obtained.

$$\frac{(\Delta t_L)_{ave}}{(\Delta t_H)_{ave}} = \frac{BER_H}{BER_{eff}} - 1 \tag{2.2}$$

As stated, BER_{eff} is approximately 10^{-6}. Let BER_H be 10^{-2}. Then Equation 2.2 shows that the ratio of high BER time to low BER time is large. Since $BER_{eff} >> BER_H$ Equation 2.2 can be simplified as shown in Equation 2.3.

$$(\Delta t_L)_{ave} \approx (\Delta t_H)_{ave} \cdot \frac{BER_H}{BER_{eff}} = 10,000(\Delta t_H)_{ave} \tag{2.3}$$

Equation 2.3 illustrates that the majority of the time, on a wireless link, the data is transmitted correctly . Only short bursts of data are corrupted but they are corrupted almost entirely. This is significant in the context of a TCP link because just a few segments (out of many) get "lost" (corrupted), but there is a certain amount of regularity to the lost segments. Figure 2.9 shows how TCP handles the lost packets for a wireless link (adapted from (18)). Because of the lost packets, the effective data rate can never be optimum.

Consider a wireless link with a data rate of 2 Mbps. For sake of simplifying calculations, the actual data rate is assumed to be 2.048 Mbps. For traditional wired networks, upon link start-up the TCP flow control mechanism restricts the data throughput. As time progresses and the link remains stable (congestion free), the flow control allows the throughput to increase and eventually reach the maximum output rate of 2 Mbps. This is illustrated by the plot in Figure 2.10. The time axis in Figure 2.10 is given in discrete

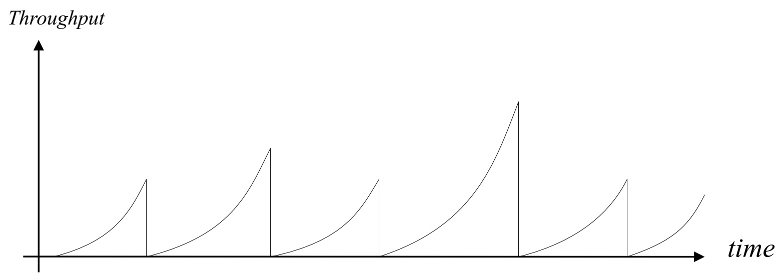


Figure 2.9 TCP Throughput for Lossy Link

time steps. These time steps determine when the TCP flow control mechanism allows the throughput to increase. This example is adapted from (113).

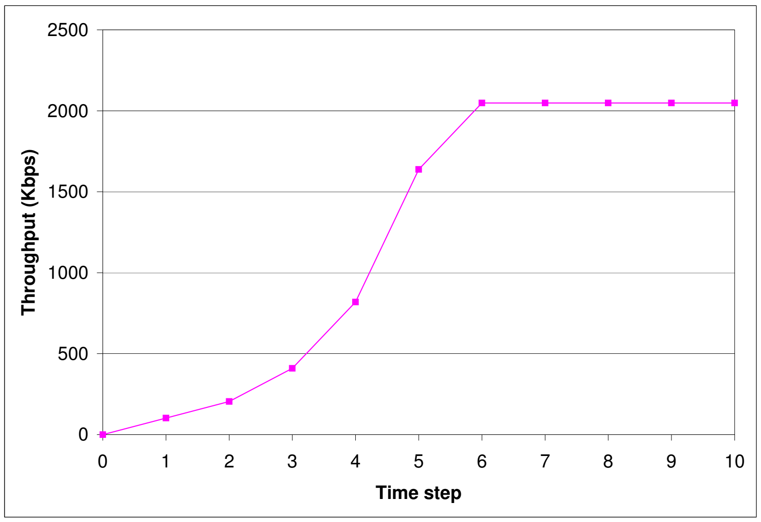


Figure 2.10 Ideal TCP Link Throughput

If bit errors cause segments to be received incorrectly, TCP assumes that the lost segments are due to congestion and begins congestion avoidance. The following analysis shows the effect of various values for $(\Delta t_L)_{ave} + (\Delta t_H)_{ave}$ related to the link characteristics. It is assumed that over a sufficient amount of time the link reaches steady state and throughput can be modeled as a periodic function. This allows the effective throughput to be calculated from Equation 2.4.

$$Throughput_{eff} \equiv \frac{1}{T}\int_0^T f(x)dx \tag{2.4}$$

where $T = (\Delta t_L)_{ave} + (\Delta t_H)_{ave}$

The function $f(x)$ is the portion of the graph shown in Figure 2.10 over the applicable time period. Since $(\Delta t_H)_{ave}$ is very small compared to $(\Delta t_L)_{ave}$, it is assumed that burst errors affect, at most, two segments. However, because of retransmissions, it is further assumed that only one segment is effectively lost. Since TCP assumes that lost segments are a result of congestion, it "over corrects" for wireless links. Ideally, on a wireless link, only the lost segment(s) needs to be retransmitted, and the congestion window should not be reduced (since the loss was not due to congestion). The steady state, periodic throughput function with a specific period is shown in Figure 2.11. The traditional TCP throughput plot is shown with an "improved" TCP version where a lost segment is simply retransmitted and the congestion window is not changed. The period in Figure 2.11, as is evident, is five time steps. Because of the discrete nature of the data, a simple spreadsheet is used to perform the "integration" of Equation 2.4.

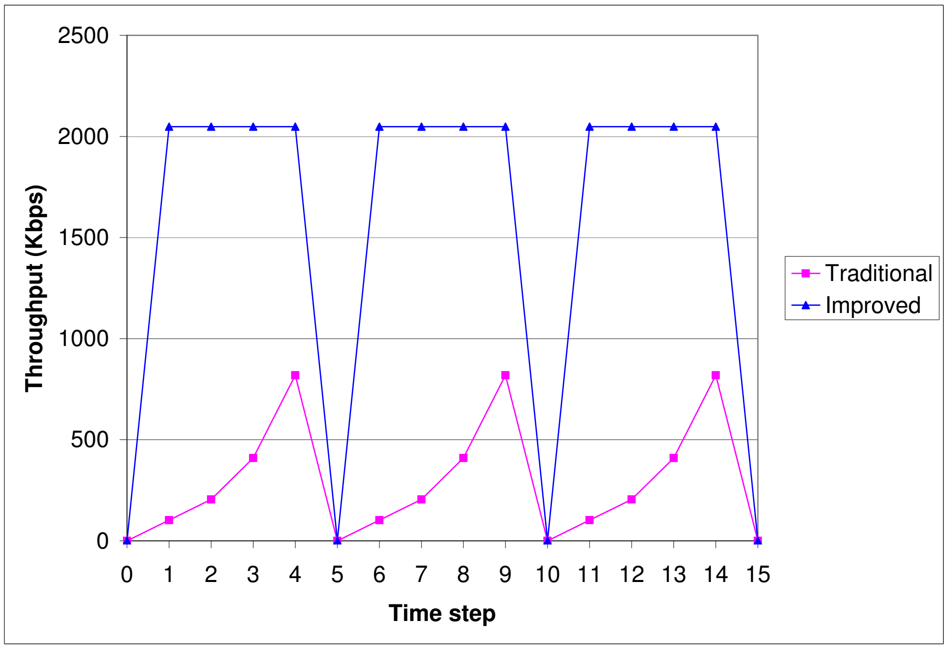


Figure 2.11 TCP Steady State Throughput

Figure 2.12 shows the comparison of effective throughput for the traditional TCP link management and an improved TCP for wireless links. The plots in Figure 2.12 are based on the assumptions and data used for the above example. It is clearly evident that some

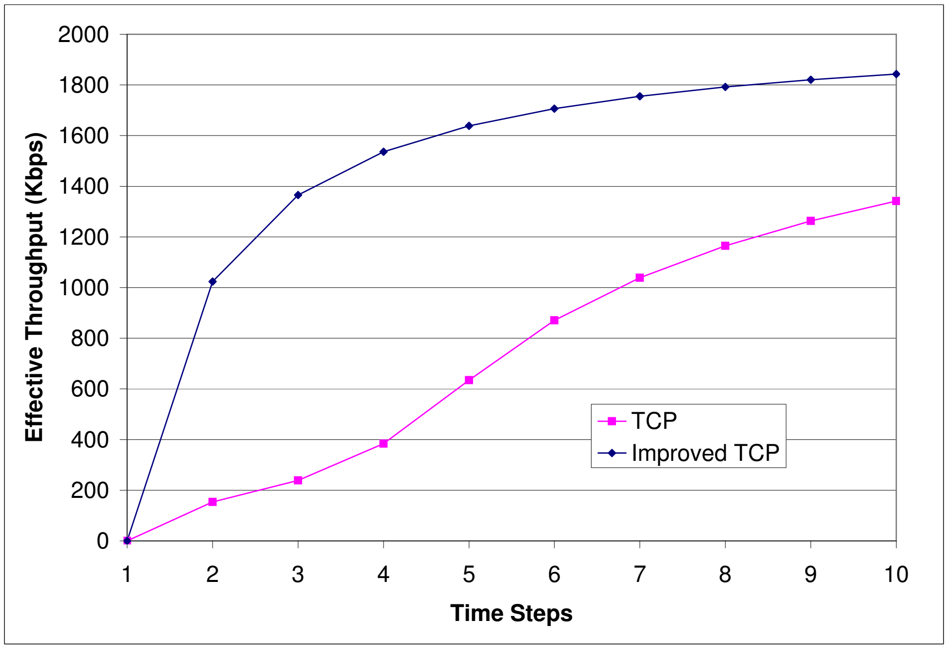


Figure 2.12 Throughput Comparison

form of a modified transport protocol is needed to more efficiently use wireless data links. In the case of wireless sensor networks, protocols that address these issues are described in Section 3.2.

2.5 Summary

The behavioral rules of swarming provide a robust mechanism for organizing large numbers of mobile sensors. The assumption of large swarm sizes requires considerable decreased sizing improvements for UAVs, miniaturization of sensors, and significant reduction in power requirements. Therefore, non-traditional solutions are required for efficient and effective use of wireless networking systems that support such mobile sensors. The Joint Battlespace Infosphere can provide a strong support system for such networked sensors, allowing for dynamic tasking that satisfies user needs in a timely fashion. The Network Embedded Software Technology effort can provide a robust computing system for the distributed sensor and computational nodes. However, the communications structure required to provide sufficient throughput for swarm applications is lacking. In order to improve the networking systems–and specifically the communications protocols–requires extensive use of simulators. The *ns-2* simulator provides a cost-effective simulations system that provides the required level of model fidelity.

III. Current Trends in Sensor Swarm Networks

As described in Chapter I, the purpose of this research effort is to develop a modeling process for swarm based sensor network applications. To that end, an algorithmic swarm model as well as development and analysis of a communications system for swarm-based sensors is needed. This chapter—providing a foundation for these developments—reviews the latest work in the areas of mobile sensor systems and the communications systems that can be used to support self-organizing systems. The first section provides a description of current and future sensor platforms. The next section describes several ad hoc sensor network designs and implementations as well as their routing schemes. The third section details the current efforts in swarm modeling.

3.1 Sensor Platforms, Today and Tomorrow

Great strides are being made in miniaturization of electronic and electro-mechanical systems (19, 67, 104) as well as investigation into non-traditional platforms (7). These devices, equipped with wireless communications systems and multiple types of sensors can provide sensor data in numerous environments including those that are unsafe for humans or unsuitable for traditional sensor systems. A large number of these devices can work together like a swarm of insects or a flock of birds to possibly provide high fidelity information on a near real-time basis.

This section describes several sensor platforms varying widely in size and capability. These platforms include the SensorCraft, Unmanned Underwater Vehicles, Smart Dust, Smart Sensor Snow, the Multipurpose Security and Surveillance Mission Platform, and the Low Cost Autonomous Attack System as well as the SkyTote.

SensorCraft. The SensorCraft is a multi-directorate Air Force Research Laboratory (AFRL) research effort (61). Headed up by the Sensors Directorate (AFRL/SN), the SensorCraft, shown in Figure 3.1, is envisioned to provide a fully integrated Intelligence, Surveillance, and Reconnaissance (ISR) capability in a UAV platform and is equipped with multiple, advanced sensing devices that are integrated into the airframe itself.

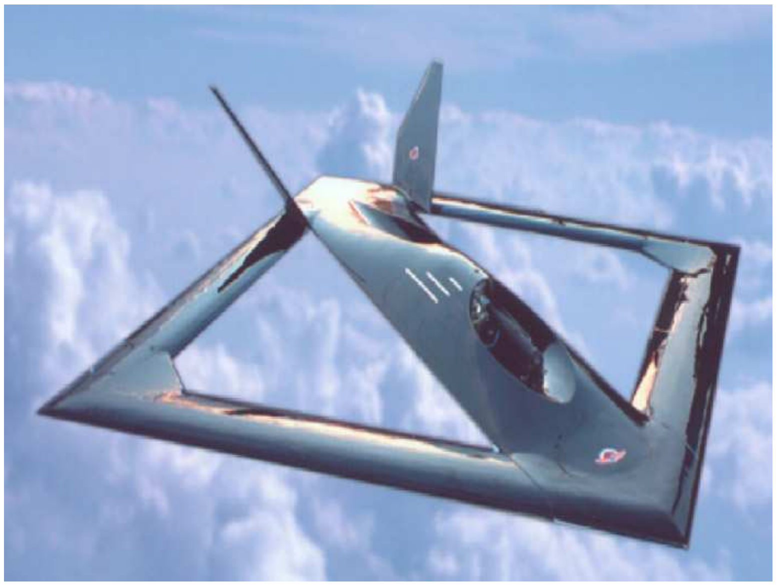


Figure 3.1 The Sensor Craft (61)

The SensorCraft is not meant to participate in a cooperative (i.e. multi-vehicle) ISR mission. The intent is to use the SensorCraft in a stand-off position as part of a heterogeneous air and space vehicle network to provide a comprehensive view of the battlefield. There are two potential problems with this scenario. First, in order to provide sufficient protection for the SensorCraft, the stand-off distance must be large. This results in less optimal sensor coverage of the area of interest. This leads to the second problem: In order to improve sensor coverage, the SensorCraft must decrease the stand-off distance (and thereby increase the risk). If a SensorCraft should be damaged or destroyed, the coverage for an entire region would be lost. This makes the SensorCraft potentially less reliable than a collection of cooperating, less expensive, less complex sensor vehicles.

Unmanned Underwater Vehicles. The Navy is vigorously pursuing an autonomous ISR capability at the Space and Naval Warfare Systems Center San Diego (SSC-San Diego) (42). The use of unmanned underwater vehicles (UUVs) is a crucial part of the Navy's ISR research and development efforts.

The Navy's collection of UUVs include the "Free Swimmer," the "Advanced Unmanned Search System" (AUSS), and the Odyssey class. The Free Swimmer system incorporates advanced technologies such as neural network controlled sensors and autonomous mission planning. While the Free Swimmer system is no longer in use, its development provided enabling technologies for the AUSS. The AUSS, shown in Figure 3.2, is an untethered UUV that is able to autonomously perform basic mission tasks such as transiting to a given location, hovering, and executing pre-programmed sonar and optical search patterns. The AUSS uses an acoustic data link to provide supervisory control and for uploading data from on-board sonar sensors and still camera images. Image data is processed and compressed before transmitting in order to more efficiently use the available data link bandwidth.

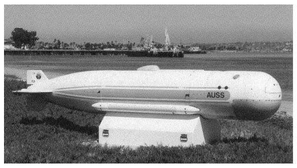


Figure 3.2 Advanced Unmanned Search System (42)

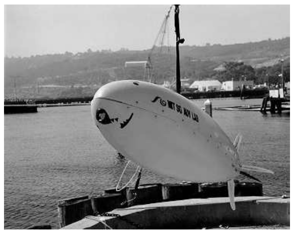


Figure 3.3 Odyssey III Autonomous Underwater Vehicle (108)

The Odyssey class of underwater vehicles is a continuing development effort of the Autonomous Underwater Vehicle (AUV) Laboratory of the Sea Grant College Program at the Massachusetts Institute of Technology (108). The Caribou, shown in Figure 3.3, is the newest of the Odyssey class of AUVs. It is capable of carrying modular sensor systems–including sonar, video, and other oceanographic sensors–to depths of 4500 meters and has an operational endurance of 20 hours (108).

Smart Sensor Snow. Smart Sensor Snow (52) is a sensor organizing algorithm for mobile robotic applications. A static, randomly distributed set of sensors self-organize in order to provide support (in terms of navigation and other location identification efforts) for mobile robots. These robots are assumed to be able to interact with the sensor network in some fashion.

The Smart Sensor Snow algorithm uses Turing's reaction-diffusion equation (106) running locally on each sensor in order to produce some desired pattern. Patterns that can be produced by this algorithm are shown in Figure 3.4. The two patterns were generated from a 200 by 200 grid of sensors using only local interaction. The first pattern makes a regular framework in which a robot can navigate while the second pattern provides a set of fixed distances from some boundary line. This can provide information to robots about distance travelled.

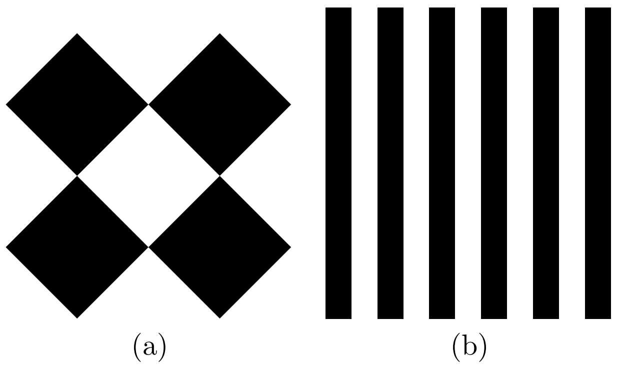


Figure 3.4 Sensor Snow Patterns (52)

The basic sensor configuration includes a rudimentary communications mechanism. The only information shared between neighboring sensors is the distance between sensors. The authors of (52) are unclear as to the exact method of the information exchange protocol, but suggest some form of uniquely identifiable "chirps" from each sensor that can be used to measure the distances between sensors. In order for the mobile robots to exploit the sensor patterns effectively each sensor must have the ability to determine or retrieve the following information elements:

- A measure of the density of sensors around it
- A measure of the distance to a sensed event
- The direction to a sensed event
- The direction toward sensor rich areas (i.e. a gradient of sensor density)
- Pattern values for various useful patterns in the application

The authors do not specify the type of sensors or the types of events which the sensors are to track. The types of applications for which this system is intended seem to be low bandwidth, event centric. Since the communications are limited to only distance information, it would be difficult to use this system for more information-rich applications.

The main computational load of this sensor network is determination of global and local frame information (i.e. coordinates in a two dimensional space). This is a simple application of cartesian coordinate calculations. It appears that the information for the global and local coordinates are propagated through the network of sensors in order to provide more locally tuned and robust robot behaviors.

Multipurpose Security and Surveillance Mission Platform. The Multipurpose Security and Surveillance Mission Platform (MSSMP) system is a distributed network of remote sensors mounted on vertical-takeoff-and-landing mobility platforms (83) (shown in Figure 3.5). The Army intends to use this system in military operations in urban terrain (MOUT). These types of operations are conducted in urban areas where manmade construction and high population densities are the dominant features (50).

Weight and power restrictions of the air vehicle and bandwidth limitations of the RF communications impose several constraints on the overall system. First, the sensor package must be small, light-weight, and low in power consumption. Second, the majority of the computations for sensory data processing must be performed on-board. This reduces the bandwidth required and the power needed to transmit. The majority of the computational load is used for image compression. The images are compressed using the JPEG compression technique (93). Some motion detection processing is also accomplished on-board. The communications architecture uses TCP/IP for internode communications. TCP/IP was chosen because it allows numerous sensor packages and control station to operate together in an Internet-like network. The authors of (83) make no mention of the underlying physical or data link layers.

The MSSMP is a small, highly mobile sensor platform vehicle that weighs less the 300 pounds, is approximately 6 feet in diameter and has a range of over 20 kilometers. The sensors used on the MSSMP include a visible light video camera, an infrared video camera,

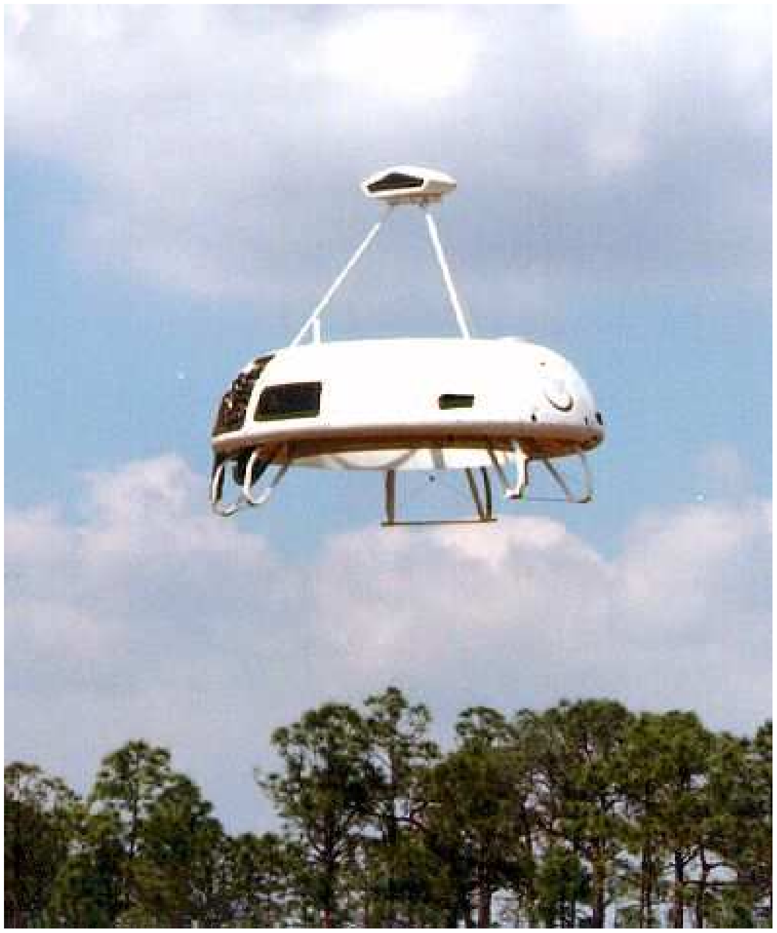


Figure 3.5 The MSSMP Air Vehicle (83)

a laser range finder, and an acoustic detector. The specific units chosen were determined by the specified requirements as well as off-the-shelf availability.

Low Cost Autonomous Attack System. While the Low Cost Autonomous Attack System (LOCAAS) (95) is not an integral part of this research activity, it is a potential platform for the swarm-based network of sensors. For this reason, the specific details and capabilities are included. A prototype of the LOCAAS is shown in Figure 3.6 and the capabilities are listed in Table 3.1. The cost of the munition is based on a 12,000 unit buy in 1994 base year dollars. The LOCAAS plays a significant role in the Smart Sensor Web which is described in Section 3.2.1.

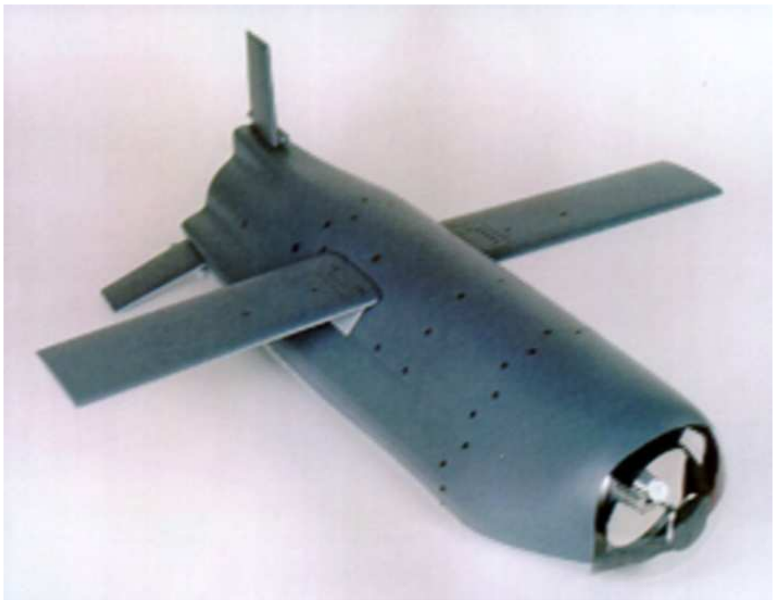


Figure 3.6 A Prototype LOCAAS (95)

Table 3.1 LOCAAS Specifications

Length	30 inches
Wingspan	40 inches
Weight	90 - 100 lbs
Engine	30 - 50 lb thrust class turbojet
Endurance	30 min. expected
Range	> 100 km
Guidance	Solid state LADAR seeker
Cost	< \$30,000

Black Widow. Developed under the DARPA Micro Air Vehicle (MAV) program, the Black Widow (2) is a small (less than six inches in diameter and weighing less than two ounces) aircraft intended for military intelligence activities. The aircraft, shown in Figure 3.7, managed to fly for 16 minutes at speeds approaching 45 mile per hour. The Black Widow was developed by AeroVironment, Inc (3) under a DARPA research contract for synthetic multifunctional materials. The MAV is expected to be equipped with a miniature video system to send visual surveillance and location information.

WASP. The WASP (35) is another of the latest DARPA MAV developments. This MAV set a flight endurance record of one hour and 47 minutes. The previous record was

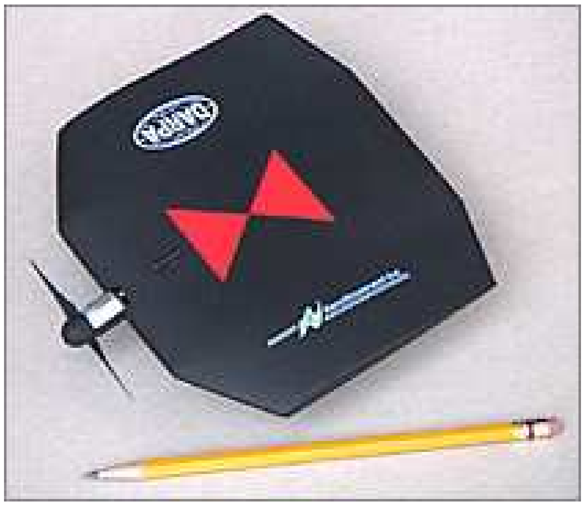


Figure 3.7 The Black Widow Micro Air Vehicle (2)

30 minutes according to a DARPA press release (35). The WASP was also developed by AeroVironment, Inc (3) under the same DARPA research program. The WASP uses electric propulsion, has a 13 inch wingspan and weighs in at just 170 grams (about 6 ounces). AeroVironment makes a variety of robotic planes and other electrically powered vehicles. Figure 3.8 shows the WASP air vehicle and Table 3.2 summarizes some of the vehicles specifications.

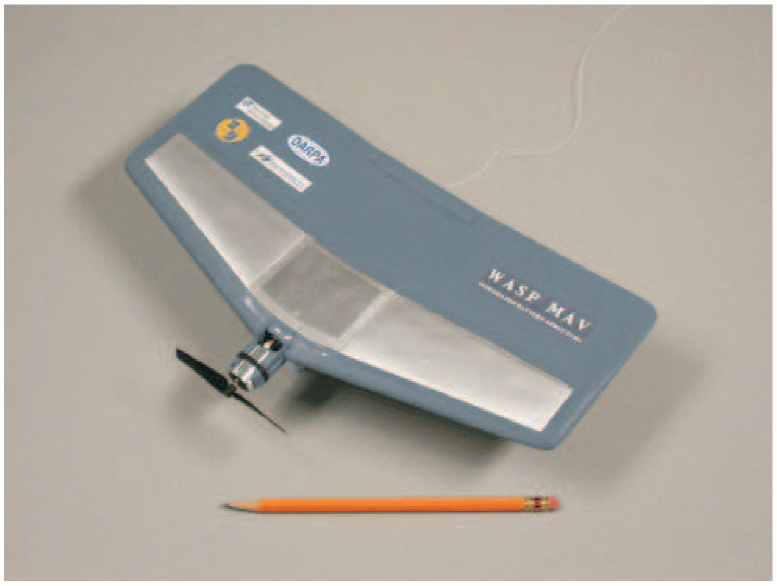


Figure 3.8 The WASP Micro Air Vehicle (35)

Table 3.2 WASP Specifications

Wingspan	13 inches
Weight	6 ounces
Endurance	1 hr 47 min

SkyTote. The SkyTote is a concept exploration effort managed by the Air Force Research Lab's Air Vehicles Directorate under a Small Business Innovate Research Phase II contract (10, 27) also with AeroVironment, Inc (3). The SkyTote has a vertical take off and landing capability and is able to transition to horizontal flight. It is designed to delivery a payload to a precise location quickly, cheaply, and safely. The SkyTote is shown in Figure 3.9. The current version of the SkyTote is remotely piloted but plans are

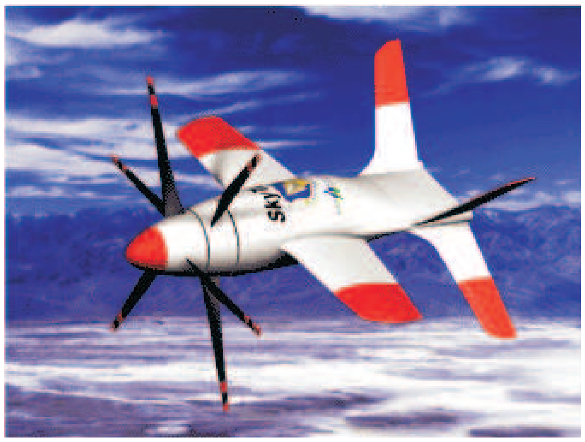


Figure 3.9 The SkyTote Micro Air Vehicle (27)

underway for autonomous control.

3.2 Sensor Networks

This section describes ad hoc sensor network implementations and their routing schemes. The *Smart Sensor Web* is a World Wide Web-like information gathering and dissemination system using low-cost, disposable sensors (11). A reconfigurable sensor network scheme, called *Directed Diffusion* (DD), provides design principles for distributed sensor applications (58). The *Geographical Addressing and Routing Protocol* (GAaRP) is part of Distributed Smart Sensor Network as described in (29).

Each of these implementations assume an ability to self-locate within the geographic environment. This is not unrealistic given the proliferation of Global Positioning Satellite (GPS) systems and their accuracy. This information can be used to great advantage as shown by the following systems. A more detailed description of GPS is included in Appendix B, Section B.5.

These ad hoc networks can be categorized in to two types: *proactive* and *reactive*. A proactive, or table-driven protocol, like the Destination-sequenced Distance-vector (DSDV) protocol, constantly sends update messages throughout the network to maintain near real-time status information at every node. Reactive protocols, like the Ad Hoc On-demand Distance Vector (AODV), build routes only when necessary (101). This type of protocol is especially useful in network environments where the physical topology is varying rapidly. The main difference between protocols like DSDV and AODV is found in the reaction time and node memory requirements. Proactive protocols provide for immediate routing of user messages at the expensive of increased background control messages and larger memory requirements on each node in order to maintain the routing tables. Reactive protocols have lower control overhead but require longer set up times to route user messages.

The DSDV and AODV routing protocols (and others discussed in (101)) are not described in further detail because of the lack of scalability. One way to improve the scalability is to impose a hierarchical addressing scheme. However, this requires the specialization of some nodes and contradicts the underlying operating principles for a swarm-based sensor network.

Other scalable routing protocol systems exist. These include the Scalable Location Update-based Routing Protocol (SLURP) (130) and the Grid Location Service (GLS) (73). Both systems use an update mechanism that maintains approximate location information of other nodes based on a partition of the physical space into regions. In order to send information to another node, the sender first queries the destination's region for the approximate location. Then a simple geographic routing protocol is used to forward the data. This is a form of a hierarchical routing protocol since two types of information must be maintained: low-level location data about nodes in a region and high-level location data about approximate (region) locations. These systems benefit from a distributed implementation in order to achieve a high degree of scalability.

The protocols described in this section typically reside at the *transport* and *network* layers of the OSI reference model (119). However, because of system limitations–including power, weight, size, etc.–the layers of the OSI model get blurred significantly in implementations.

3.2.1 Smart Sensor Web. The Smart Sensor Web (SSW) consists of low-cost, disposable sensors (see Section 3.1). The information from these sensors can be fused into an intelligence network of several *sub-webs* that are accessible at the lowest level on the future battlefield (11). A conceptual view of the SSW is shown in Figure 3.10 (adapted from (75)). The SSW is an Army initiative that is also intended for military operations in urban terrain (MOUT) environment (see Section 3.1). However, the other services are involved at varying levels. The Air Force's involvement includes the LOCAAS as part of the Weapons Web (75).

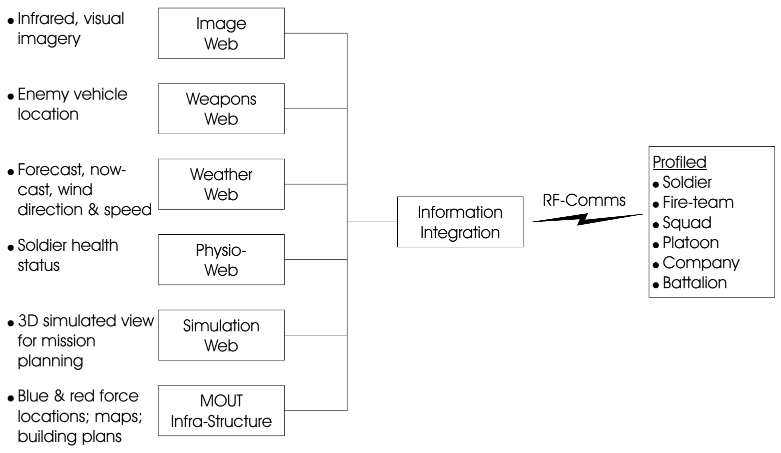


Figure 3.10 Smart Sensor Web Concept (75)

The goal of the SSW is to integrate sensor information from various sources and provide it to all levels of the command hierarchy in an appropriate manner. As shown in Figure 3.10, the link to the user is based on radio-frequency (RF) communications.

As part of the Weapons Web, LOCAAS is an ideal candidate for consideration in a distributed target identification and engagement scenario. The LOCAAS consists of a low cost laser detection and ranging (LADAR) seeker with Automatic Target Recognition and an on-board Inertial Navigation System/Global Positioning System integrated into a small, air vehicle powered by a miniature turbojet engine (95).

The SSW does not specify implementation details but rather sets a goal for a distributed system of information producers (sensors) and users. This is similar to the JBI. The implementation details are left to the designer so that the requirements can be met using an effective and efficient communications architecture.

3.2.2 Directed Diffusion. Directed Diffusion is a data-centric dissemination methodology for large scale, dynamic sensor networks (58). The authors of (58) provide the following example that illustrates how Directed Diffusion works.

> One or more human operators pose, to any node in the network, questions of the form: "How many pedestrians do you observe in the geographical region X?", or "Tell me in what direction that vehicle in region Y is moving". These queries result in sensors within the specified region being tasked to start collecting information. Once individual nodes detect pedestrians or vehicle movements, they might collaborate with neighboring nodes to disambiguate pedestrian location or vehicle movement direction. One of these nodes might then report the result back to the human operator.

In this description, an *interest* is expressed in the form of a query that is passed on to the sensor network. It is assumed that nodes have a self-locating capability so that the location of the interest relative to the location of each sensor in the potential communications path is known. This establishes a *gradient* within the sensor network. This gradient information is used to route data requests and responses within the network of sensors without the use of routing tables.

Directed Diffusion is different from the widely used Internet Protocol (IP). IP routing requires a sufficient knowledge of the network—in the form of IP addresses and routes to those addresses—in order to establish end-to-end connections for communication service requests. Directed diffusion, on the other hand, requires no global routing information in order to achieve connectivity. However, this means that not all paths found are optimal. The authors state that the benefits gained in reliability and robustness outweigh the potential disadvantages of suboptimal routing.

Some underlying assumptions are made concerning the sensor nodes and the expected architecture of the sensor network for a system that uses the Directed Diffusion routing algorithms. A summarized list of these assumptions is given in Table 3.3.

Table 3.3 Directed Diffusion Sensor Configuration (58)

• Matchbox sized form factor
• Battery power source
• Power efficient microprocessor
• Several megabytes of program and data
• RF modem using some form of diversity coding
• Energy efficient Media Access Control sublayer
• Stripped down version of a modern OS (e.g. Windows CE or μCLinux
• Multiple sensors per node
• Fully functional Global Positioning System receiver

Traditional sensor systems fall into two broad categories. The first category includes those sensor systems that are large and complex and are deployed at great distances from the phenomena to be observed. This requires complicated processing to overcome the noise and interference associated with the large distances involved. The SensorCraft (Section 3.1 above) is a prime example of such a sensor system. The second category is made up of systems in which less complex sensors are laid out in a carefully engineered pattern in the region to be observed. Because of the deterministic placement scheme, the network communications topology can be designed to the specific implementation of the sensor system. Additionally, these sensors, because of their reduced capability, must send a continuous series of data samples to one or more central processing nodes for data analysis (reduction and filtering).

The problems with the first category are the same as those for the SensorCraft. Too much capability in a single node makes the node too valuable to risk. This degrades the ability of the sensor system to collect high fidelity data because of large stand-off distances. The problem with the second category is two-fold: First, a large amount of up-front engineering is required to determine the best configuration for sensor deployment. This is offset somewhat by the fact that the network topology can be optimized for the specific application. Second, because of the need for continuous sampling, bandwidth resources may become overwhelmed thus causing data to be lost.

DD Protocol Description. As stated, an *interest* is used to task the sensor network to begin sampling the environment for some type of event. The type of event is dependent on the particular sensors used in the network. Once a sensor receives the tasking it begins reporting responses in the form of data messages. An interest message is input into the sensor network at a *sink*. A sink is any node that receives the interest message. The sink node remains constant for the duration of the interest task. In other words, all responses to the interest tasking are routed back to the original sink node. This is significant since it implies that the network of sensors is not dynamic.

A notional network is shown is Figure 3.11. The network has a single sink node (indicated by the double black circle) and several other regular sensor nodes. The dashed line rectangle represents the tasking subregion for the interest.

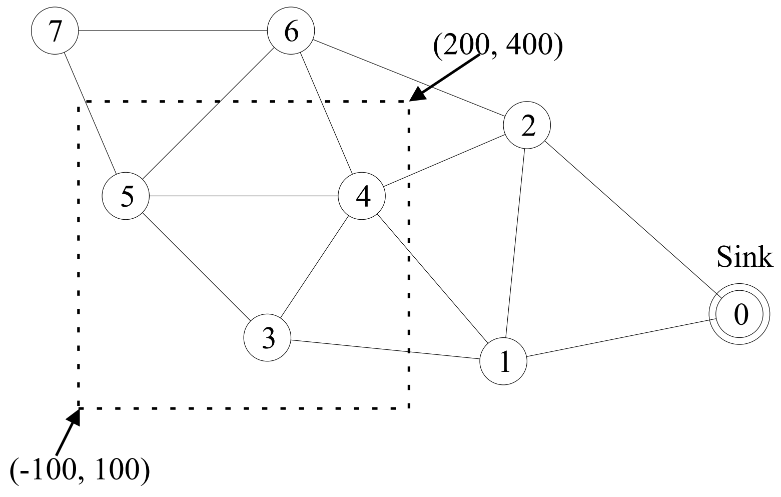


Figure 3.11 Directed Diffusion Network Example

An example interest would be formatted as shown in Figure 3.12. The `type` field is based on some classification of the objects of interest to the sensor network. Classes might consist of `soldier`, `truck`, `tank`, or `car`. The `interval` specifies the sampling rate at which events are to be reported. The `duration` specifies how long the *interest* is valid. Finally, the `rect` field specifies the subregion in which events are to be tracked.

```
type      =  tank
interval  =  20 ms
duration  =  10 minutes
rect      =  [-100 100 200 400]
```

Figure 3.12 Sample *Interest* Specification

The `rect` field, in this example, is given in some form of global coordinates but it could easily be specified using GPS coordinates.

The *sink* node in Figure 3.11 initially receives the interest from some outside source and saves the interest message in its *cache*. This cache is used to for several purposes. First, it ensures that loops within the network do not cause unlimited propagation of a single interest. Second, it serves to track received data messages so that *gradients* can be updated. These gradients *draw* data from the sensor field in such a way as to reinforce better (shorter, more reliable) routes through the sensor network. It should be noted at this time that the communications mechanism must provide some unique identification scheme for each node. Such unique naming schemes as those used for the 802.11 machine access control addresses or the Bluetooth cluster addresses are good examples (see Appendix B for more details).

Once the sink has processed the interest and stored it in its cache, it forwards a slightly different message to its neighbors. In the example of Figure 3.11, the sink neighbors are nodes `1` and `2`. The interest message that is forwarded specifies a much longer `interval` field. An example of this message is given in Figure 3.13.

```
type       =  tank
interval   =  1 sec
rect       =  [-100 100 200 400]
timestamp  =  10:20:00
expiresat  =  10:30:00
```

Figure 3.13 Propagated *Interest* Specification

Since the network nodes can only use local information, the longer interval allows for exploration of the network without flooding the network with interest messages. The exact value for the slower data rate specification is a matter of further research(58). The choice represents a trade-off between faster interest message propagation versus network flooding. Additionally, since message receipt is not guaranteed, the interest message is repeated at regular intervals.

The nodes store the modified message in their cache in the following way. For each interest an entry is made in the cache. Interests are uniquely identified by their `type`, `interval`, and `rect` fields. Each entry in the interest cache contains several fields. In addition to the fields described above, there is a `gradient` field with an entry for each neighbor from which it received an interest. The `gradient` field contains entries for the `data rate`, `timestamp`, and `expiresat` fields. Once the nodes store the message it is forwarded to their neighbors. Since each node can determines its position, it can decide whether the interest is a task for its sensors or not. In the example, nodes `3`, `4`, and `5` are in the subregion of interest. Once the interest has *diffused* through the network, the nodes in the subregion of interest wait for an appropriate event. The cache entries of each node provides a "snapshot" of the connectivity of the network at the time the interests are propagated through all of the nodes. For instance, the interest cache for node `4` appears as shown in Figure 3.14.

```
type = tank
interval = 1 sec
rect = [-100 100 200 400]
timestamp = 10:20:15
duration = 9:45
gradient (i, r, d)
    (1, 1, 9:55)
    (2, 1, 9:55)
    (3, 1, 9:50)
    (5, 1, 9:45)
    (6, 1, 9:50)
```

Figure 3.14 Interest Cache Entry for Node 4

The `gradient` entries are *node ID*, *update rate*, and *duration* respectively. A delay of 5 seconds was used in calculating transmission times. This exaggerated value is useful in illustrating the *diffusion* of the interest message through the network. The entries in the cache for node 4 are interpreted as follows. For any event (data message) that matches this interest, the data message is to be transmitted to each node according to the data rate specified in the gradient field for that node. In this example all the entries are 1 so that each neighbor node is to receive data messages at the same rate. But as is shown, the data rates change according to the performance of the network over time. The data rates are increased for paths that are better (shorter, faster data rates, etc.).

To continue the example, let an event that matches the specified interest occur in the proximity of node 4. This event generates a data message of the form shown in Figure 3.15

```
type        =  tank
instance    =  M1A1
location    =  [180 310]
intensity   =  0.6
confidence  =  0.85
timestamp   =  10:22:05
```

Figure 3.15 Sample *Data* Message

In this example the tank detected is determinied[1] to be an Abrams M1A1 Battle Tank. The `location` field contains the coordinates of the sensor node that recorded the event. The `intensity` and `confidence` fields are based on the sensor signal strength and the target recognition algorithm. Finally, the `timestamp` field records the time of the event.

The data message is sent by node 4 according to the entries in its interest cache. In this case, data messages are sent to nodes 1, 2, 3, 5, and 6 at the same rate. As the data message propagates through the network, it is cached at each node in a data cache (separate from the interest cache).

In an ideal environment, the messages sent to nodes 1 and 2 clearly reach the sink node before all other data messages propagate through the network. In fact, assume the message from node 4, through node 2, reaches the sink first. The sink uses this information to *reinforce* node 2. This is done by increasing the data rate—say to 2 samples per second (an interval of 0.5 seconds). The sink then reissues the interest with the new, increased

[1]Object identification is accomplished through some form of automatic target recognition such as MSTAR (71)

sample rate but only to the neighbor from which it received the data first (in this case node 2). When node 2 receives the new interest message, it updates its cache entry and examines its data cache to see what nodes it received data messages from that match the specified interest. The node that sent that data message first is reinforced (in this case node 4). The interest cache for node 4 is updated as shown in Figure 3.16. In this way,

```
type = tank
interval = 0.5 sec
rect = [-100 100 200 400]
timestamp = 10:22:25
duration = 7:35
gradient (i, r, d)
    (1, 1, 7:25)
    (2, 0.5, 7:25)
    (3, 1, 7:20)
    (5, 1, 7:15)
    (6, 1, 7:20)
```

Figure 3.16 Updated Interest Cache Entry

the path from node 4, through node 2, to the sink is reinforced until the data rate matches the original rate specified by the initial input interest query data interval of 20 ms.

Since the other data rates (*gradients*) at node 4 are never decreased, data is still transmitted via the other routes. This represents an overhead of spurious messages as long as the route from node 4 through node 2 to the sink is providing an acceptable transmission rate. However, if for some reason one of the links were to be broken (because of node failure at node 2 or some type of RF interference), the route through node 1 would take over (at the initial slow rate of 1 sample per second). Since the sink would begin receiving data from node 1 at a higher rate, it would reinforce node 1 and begin to degrade node 2. This mechanism provides the robustness of the sensor network at the cost of redundant message traffic.

While not mentioned here specifically, the authors in (58) describe how the sensor network can aggregate information from multiple sensors in order to provide better location information or a better confidence measure. This is easily seen since each node caches any data messages that it receives.

The shortcoming of this algorithm is the requirement that the nodes, once placed, remain fixed. Since each node has a GPS receiver, it could use its location to provide a unique *location based* addressing scheme. Then, gradients could be used in the more traditional sense—information would flow *downhill* toward the sink. Care must be taken to provide reliability in the form of alternate paths. But the data rate, in conjunction with a geographical based addressing scheme, could be used to provide for low data rate redundant messages along less optimum routes. This would ensure that dynamic re-routing is capable of providing the necessary multiple links. A form of this proposed routing mechanism is used in the routing protocol described in the next section.

3.2.3 Geographical Addressing and Routing Protocol. The Geographical Addressing and Routing Protocol (GAaRP) is also intended for ad hoc sensor networks in a highly unstructured physical configuration of nodes (28, 29). In contrast to Directed Diffusion, GAaRP allows for mobile nodes. Routing within the network is based solely on a node's location as specified by an onboard GPS receiver.

The GAaRP routing protocol is designed to support the Distributed Smart Sensor Network (DSSN). The DSSN is made up of small, inexpensive, wireless units. Each unit has a rudimentary microprocessor, one or more sensor elements, a GPS receiver, and an RF transceiver. The sensor node design is intended to be modular in order to facilitate component upgrades or sensor element changes without the need to redesign the entire system.

The DSSN is made up of multiple sensor nodes and *Home* and/or *Gateway* nodes. The *Home* node is defined as the location where a user is allowed to interact with the sensor network. A *Gateway* is an access point of some local area network to the DSSN. Also, deployed sensor units can be in one of two states—*Module* or *Hub*. The topology of the network at any given time determines the state of each module but modules are allowed to switch back and forth between states indefinitely.

The DSSN assumes some form of transmission packet formatting and processing. While not immediately clear, it appears the DSSN uses a time division multiple access scheme like slotted Aloha (119) to manage contention for the communications medium. Communications take place on a single RF frequency. Therefore, some form of collision detection or collision avoidance must be used.

Each sensor node is assumed to have some of the same characteristics as described earlier: the nodes are battery operated so the algorithm and components must be designed/chosen in order to minimize power consumption, the nodes have limited memory and processing capabilities so that only small amounts of data and simplistic algorithms can be used, and RF transmissions have limited range so communications must be accomplished in a hop-by-hop manner.

The design of a sensor network protocol can take advantage of the specialization associated with the sensor application domain. The GAaRP routing protocol is no exception. The assumptions presented in Table 3.4 are made in order to simplify the protocol (29).

Table 3.4 GAaRP Network Characteristics (29)

1.	The network is used for data acquisition, not for communication in the traditional sense
2.	The network consists of many small sensor units with *one* or *few* central *Home* terminals
3.	When collecting data from the network, a user only cares about *where* the data came from and not *who* it came from
4.	All sensor data is localized (this implies that no aggregation is allowed)
5.	The network is polarized (i.e. data flows only one way—from sensor units to *Home* while control information flows only from *Home* to sensor units)

GAaRP Protocol Description. The authors of (29) describe the protocol as being able to interface with two or more *Home*/*Gateway* terminals but do not provide a description of this mechanism. Therefore, the description of the algorithm presented here focuses on a single user access point (via either a *Home* or *Gateway* node). Multiple

Home/Gateway terminals are probably handled with multiple data sets (akin to multiple entries in routing tables).

Initially, the sensor nodes are assumed to be randomly placed but within the range of at least one other node. Network topology routing is determined using a dialogue-based protocol that starts with the *Home* node and propagates through the sensor nodes. A cost function is used to evaluate multiple paths. However, since only local information is used, the final routes may not be optimum (in terms of hop count and bandwidth resource utilization).

The *Home* node begins the processing by broadcasting a *Spark* message. This, in effect, "wakes up" the nodes that are within range and causes them to begin to establish communication links with the *Home* node. The sensor units do this by broadcasting a *Who's out there?* (WOT) message. Only nodes that are "awake" can respond to the WOT message. The response comes in the form of a *Who's out there acknowledgment* (WOT-ACK) message. The WOT-ACK message contains a Cost Function Value (CFV) from the responding unit. The neighbor with the "best" CFV is assumed to be the optimal choice for a route to the *Home* node according to the current network topology. The CFV is calculated using the formula in Equation 3.1.

$$CFV = C_1 D_{radial} + C_2 A_{vector} + C_3 N_{hops} + C_4 N_{slots} \tag{3.1}$$

D_{radial}	the absolute distance between nodes
A_{vector}	the delta angle vector between the neighbor node and the *Home* node
N_{hops}	the number of hops between a neighbor node and *Home*
N_{slots}	the number of available packet slots allocated to the neighbor node

The authors of (29) state that the constants C_1, C_2, C_3, and C_4 are weights based on the topology of the network. The exact meaning of this is unclear since the network is allowed to be dynamic. However, these constants may be chosen based on some physical characteristics of the nodes (in terms of RF transceiver range, sensor footprint, etc.). The meaning of the parameters in Equation 3.1 is further illustrated in Figure 3.17.

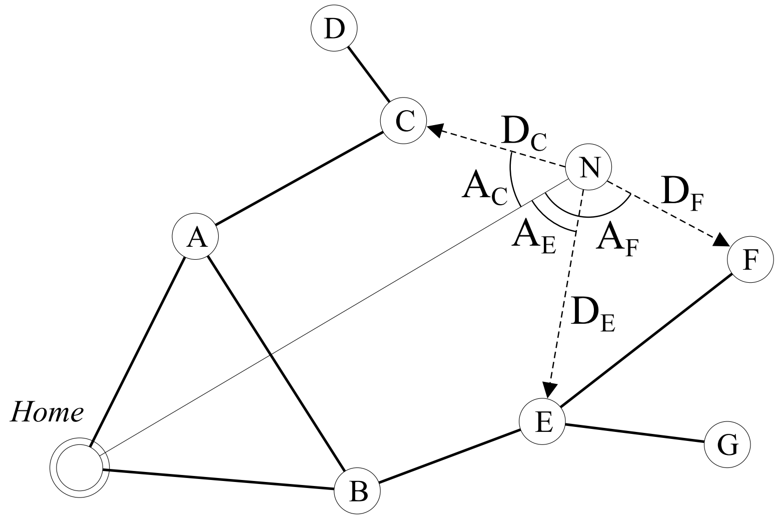


Figure 3.17 DSSN Network

The new node being inserted into the network is node N. In order to begin the negotiation process, node N must have received a *Spark* message from nodes C, E, or F. For the example of Figure 3.17, the distances between node N and nodes C, E, and F are D_C, D_E, and D_F respectively. The angle vector parameter is also calculated as shown. Since the optimum direction would be directly toward the *Home* node, the angle specifies the amount of departure from that optimum value for each potential route. The number of hops and slots (N_{hops} and N_{slots} respectively) are not shown. The vectors between the nodes can be calculated based on the underlying two dimensional coordinate system. For instance, the vector pointing from node N to node C can be found as shown in Equation 3.2.

$$\boldsymbol{V}_{NC} = P_C - P_N \tag{3.2}$$

where P_C and P_N are the locations of nodes N and C respectively. The vector from node N to *Home* can be found in a similar fashion. The parameters D_{radial} and A_{vector} between node N and its neighbors can then be found using these vector quantities. The radial distance is given by the Euclidean norm of the vector $\boldsymbol{V}_{NC}$ as shown in Equation 3.3.

$$\begin{aligned}(D_{radial})_C &\equiv \|\boldsymbol{V}_{NC}\| \\ &= \sqrt{(x_C - x_N)^2 + (y_C - y_N)^2}\end{aligned} \tag{3.3}$$

The angle between the vector to the *Home* node and the potential route nodes can be found using the vector dot product as shown in Equation 3.4.

$$(A_{vector})_C \equiv \cos^{-1}\left(\frac{\boldsymbol{V}_{NC} \cdot \boldsymbol{V}_{NH}}{\|\boldsymbol{V}_{NC}\|\|\boldsymbol{V}_{NH}\|}\right) \tag{3.4}$$

The N_{hops} parameter is simply a count of the number of links between each neighbor node and the *Home* node (for the shortest path). The N_{slots} parameter is a function of the underlying communications management scheme but is directly linked to the number of incoming connections–this detail is not included in (29). Table 3.5 summarizes the possible values for each of the potential route nodes for the new node `N`.

Table 3.5 Cost Function Values

Node	D_{radial}	A_{vector}	N_{hops}	N_{slots}
`C`	10.2	41.7	2	1
`E`	15.6	48.6	2	2
`F`	11.5	116.8	3	0

The units of D_{radial} are based on the underlying coordinate system. The distance could be specified in yards, meters, miles, etc. The exact choice is irrelevant as long as its use is consistent. The units for A_{vector} are assumed to be degrees although radians could just as easily have been used. Therefore, the "best" CFV is the smallest. Based on this discussion, the best neighbor for the new node to connect with is node `C` (this assumes the constants of Equation 3.1 are set to unity).

Returning to the algorithm description, the new node `N` broadcasts the WOT message and waits for replies. The nodes that hear the WOT message respond with a WOT-ACK message that includes the parameters required to calculate the CFVs at node `N`. The WOT/WOT-ACK messages are exchanged several times to ensure that all possible nodes have been heard from. The authors of (29) have empirically determined that five rounds of WOT/WOT-ACK messages are sufficient[2]. Once the new node has "selected" the best potential neighbor routing node, it broadcasts a *Be My Hub?* (BMH) message to that node. A node receiving the BMH message checks to see if it has sufficient resources to support the new node and responds with a "yes" message if it can or a "no" message if it cannot. If the reply is "no" then the new node gets the next potential neighbor node (in its list of nodes ordered by their CFVs). The "yes" message from the neighbor node generates a *You Are My Hub* message to the neighbor node and causes the new node to adjust its parameters (N_{hops}, N_{slots}) accordingly.

Once the new node is successfully integrated into the network it deletes all the data it received as part of the WOT-ACK messages. The only data that must be maintained is the parameters required to calculate the CFV. In this way no global routing information is required to route data through the network. This node can then generate its own *Spark* message for other nodes not yet a part of the network. It should also be noted that a node cannot reply to a WOT message unless it is already a part of the network. This causes the network to be "built" in a radial pattern starting with the *Home* node.

Changes to the physical network topology (because node mobility) is handled by repeating the WOT/WOT-ACK process. Once a node loses connectivity with its established hub neighbor, it begins broadcasting periodic WOT messages. This causes the above process to be repeated in its entirety. The ability to detect loss of connectivity implies some network status checking which is part of the overhead of the network system. Some of it can be handled simply by keeping track of data messages. However, if data messages are not occurring at regular intervals, then some other mechanism must be available to monitor connectivity status.

In a highly dynamic environment, the repetition of the WOT/WOT-ACK process for "lost" nodes could potentially overwhelm the network and prevent timely and efficient propagation of sensor data to the *Home* node. A more loosely restrictive protocol,

[2]It appears from (29) that this parameter is independent of the number of neighbors. However, it seems that the authors assume a certain minimum distance between neighbors which would serve to limit the number of neighbors.

similar to the Directed Diffusion paradigm, would decrease the overhead associated with reestablishing communications links.

3.2.4 Other Wireless Network Protocols. As stated in Section 2.4, there are numerous network routing protocols for mobile ad hoc networks (see (101) for an excellent overview) as well as several efforts at the Air Force Institute of Technology (9, 107, 120). These protocols fall into two general categories: *table-driven* and *source-initiated*. The Destination-Sequenced Distance-Vector (DSDV) and Clusterhead Gateway Switch (CGS) routing protocols are examples of table-driven protocols. Each node in the network is responsible for maintaining global connection information in the form of routing tables. This requires periodic broadcasting of control messages to maintain and update the routing tables. The overhead for these types of protocols comes in two forms. First, there is a significant amount of network traffic to maintain the tables. Second, memory is required on each node to store the routing table information. The advantage of these types of protocols is the speed with which routes can be established. Since every node contains an up-to-date routing table, user messages can be transmitted without delay.

The second general protocol category is source-initiated. The Ad hoc On-demand Distance Vector (AODV) and Temporally Ordered Routing Algorithm (TORA) routing protocols are examples of protocols in this category. These protocols establish routes only as needed through the use of some form of *route request* and *route reply* messages. These protocols do not suffer from the background network traffic and memory requirements of the table-driven protocols but result in longer delays when user data needs to be transmitted. The delays are reduced by *caching* information about previously established routes (101). However, in a highly mobile network, these caches could quickly be outdated.

This section describes only a few of the numerous wireless network protocols that have been developed. They are mentioned here for completeness–these types of protocols are not considered in this research because of the excessive overhead (in terms of the richness of the network layers) required for implementation.

3.3 Swarm Modeling

The previous section describes the challenges and the technical approaches to overcoming these challenges for wireless, ad hoc networks. This section provides a review of current swarm modeling efforts. Chapter IV is a detailed description of the swarm model used in this research effort.

The swarm modeling research described in (121) focuses on particle swarm simulation for the purposes of modeling the interactions of plasma particles and electromagnetic fields. The authors adapted the particle-in-cell (PIC) or particle-mesh method (49) of modeling swarm dynamics to flights of UAVs and fighter aircraft. The advantage of PIC codes is that they scale linearly with respect to the number of particles. However, since it is mesh– or grid–based, it requires a discretization of the swarm environment. This reduces the accuracy of the model by increasing the granularity. Further, the simulations referenced in (121) are limited to small swarms (≈ 20 particles).

From the applications domain, (25) describes the challenges of autonomous vehicle control for the aerospace industry. The author describes six functions for which swarms of UAVs might be used. These functions are listed in Table 3.6. Most of these functions

Table 3.6 UAV Swarm Functions (25)

• Area search and attack
• Surveillance and suppression
• Psychological warfare
• Diversion
• Software reduction
• Survivability Enhancement

are self-explanatory. However, “Software reduction” deserves further explanation. This aspect of swarming is intended to highlight the fact that swarming applications require less software to be coded for control systems. Making use of biologically inspired control system concepts may drastically reduce the software costs of future control systems (25).

Mataric, in (77), develops a swarming system for a set of physical robots. The robots consist of a 12-inch long steerable car base equipped with a suite of infra-red sensors for collision avoidance and object detection, micro switches and bump sensors for contact detection and radios and sonar sensors for triangulating their position relative to two stationary beacons. These beacons broadcast small messages within a limited radius.

The significance of their research is that it is one of only a few research efforts in the swarming domain that uses physical robots. Additionally, for a physical system, the number is quite large (more than 20 units were used in several experiments).

The major disadvantage is the requirement of the stationary beacons. This arrangement may work for an application that is confined to a limited space, however, for applications that encompass hundreds or even thousands of square miles over potentially harsh terrain, it is not possible to place enough beacon stations.

Swarming strategies are developed in (20) and (21) for the purposes of minefield clearing. The strategies described include *random movement*, *relay clustering*, *flocking*, *swarming*, *formation maintenance*, and *comb movement*. These strategies, or behaviors, are obtained via a vectorial movement scheme. This method uses four vectors for each robot to generate a new direction. The vectors are based on obstacles, goals, position maintenance, and direction maintenance. The swarm simulation tool developed in this work is based on the capabilities of the Khepera robot (64). It is clear from this work that only small numbers of robots are simulated (four squads with 4 robots in each squad) and is not scalable.

A detailed description of a possible swarm taxonomy is presented in (37). This work, while not presenting any swarming algorithm, provides a useful mechanism for classifying swarms (or collectives) according to communications and computational capabilities. Classifications are made by size, communications range, communications topology, communication bandwidth, collective reconfigurability,

Table 3.7, reproduced from (37), provides the classification categories and a short description of each. These classification categories, along with several additional behaviorally based categories, are used later in Section 6.2.2 in categorizing swarm behavior with respect to swarm-based sensor network. Specifically, the performance of routing protocols are related to the categories of size and topology.

Table 3.7 Swarm Classification Categories(37)

Category	Description
Collective Size	The number of autonomous agents in the collective.
Comm. Range	The maximum distance between two elements of the collective such that communication is still possible.
Comm. Topology	Of the robots within the communication range, those which can be communicated with.
Comm. Bandwidth	How much information elements of the collective can transmit to each other
Collective Reconfigurability	The rate at which the organization of the collective can be modified
Processing Ability	The computational model utilized by individual elements of the collective
Collective Composition	Are the elements of the collective homogeneous or heterogeneous

A slightly different algorithm is proposed in (32). The basic principles of alignment, cohesion, and separation are the same. However, the alignment rule is applicable regardless of whether a particle is too close or too far. This is illustrated in Equation 3.5. Cohesion and separation are encapsulated in the $\boldsymbol{v}_{attract}$ vector. The sign determines whether particles are attracted or repulsed. The weights α and β can be functions of the distance between two particles as well as other parameters (speed, direction, etc.).

$$\boldsymbol{v}_{resultant} = \alpha_{attract}\boldsymbol{v}_{align} + \beta_{attract}\boldsymbol{v}_{attract} \tag{3.5}$$

It should also be noted that only particles within some maximum distance are considered for these calculations. This is a result of the *locally observable phenomena* characteristic. This maximum distance is another parameter that determines the characteristics of a swarm formation. This computational approach is used to develop a model of particle swarm movements. A more detailed mathematical description of these interactions is given in Chapter IV.

A comprehensive review of the other swarm algorithms presented in literature (37, 77, 121) revealed that (32) captures the *results* of swarm particle interaction. These *results* relate swarm member perceptions to the amount of reaction, i.e. the relatively greater importance of a nearby particle to that of a more distant particle member. Additionally, (32) provides a basis for a parameterized algorithm for behavior analysis.

3.4 Summary

This chapter provides an in-depth presentation of the issues concerning mobile sensor systems, the network communications systems that connect them and swarming algorithms that can be used to coordinate their movements. Among sensor systems, the SensorCraft (61) is the least suited to swarm based sensor applications. Its expense and complexity make it a poor choice for a large population of sensor vehicles. On the other hand, the LOCAAS (95), SkyTote (10), and WASP (35) vehicles are well qualified for use in swarming applications. Their relative low cost and simple designs make them a good vehicle choice. However, the significant differences in performance characteristics–speed and endurance–mean that the specific applications for each vehicle are of necessity different. For instance, the LOCAAS with its greater speed and lower maneuverability make it more ideal for finding and attacking fast, well-identified targets. The WASP, with its slower speed, greater maneuverability, and longer endurance, make it more ideal for surveillance and reconnaissance activities. The vertical take off and landing capability of the SkyTote make it well suited to a much wider range of missions.

By far and away, the Directed Diffusion communications protocol (38, 58) is the most predominantly used system for networking mobile sensors. It provides efficient data transfer with little overhead and is easy to implement. It adapts well to network dynamics including node failures. However, its use has been limited to small networks with little or no mobility. The research presented in this dissertation shows its applicability to swarm-based sensor networks.

The work conducted by Reynolds (99) established the foundation for swarming algorithms. Further work by others such as Crombie (32), Dudek (37), Mataric (77, 78), and Trahan (121) have extended the foundation. However, they did not provide a mechanism whereby dynamic swarm behavior can be investigated, analyzed, and categorized. The research presented here takes swarm behavior analysis to the next level by providing a methodology for investigation of behavioral dynamics.

IV. Swarm Modeling Methodology

Swarm behavior, as manifested by biological organisms, provides an effective model for developing a system for mobile sensor platforms. Flocks of birds, schools of fish, and swarms of insects demonstrate properties that are "ideal" for networked mobile sensor systems. The complete set of properties of these biological systems consist of swarm *cohesion*, as well as particle *avoidance* and *alignment* (see Table 2.1 and Figure 2.5). These properties, when implemented in a swarm-based sensor network system, provide region coverage (through avoidance), connectivity (through cohesion), and fault tolerance (through formation adaptation). However, the network dynamics challenge traditional routing protocols (as described in Section 2.4.3) due to the rapid changes to the communications topology.

It is important to note that the term *swarm behavior*, as used in here, is different than the *behaviors*, or *rules*, described in literature (99). In the context of this investigation, *swarm behavior* refers to the globally observable formation characteristics of the swarm. This is different than the rules that are enforced with varying weights within the swarm to achieve cohesion, alignment, and separation which relate directly to individual behavior.

This chapter presents the mathematical swarm model that is used to investigate swarm behavior. The model, evolved from (32) with innovative additions for peripheral vision and blocking, is described in detail in Section 4.1. Parameters that affect behavior are described in Section 4.3. These parameters are discussed in the context of steadiness, behavioral states, movement closure, and neighborhood sizes. These concepts are used as a foundation for a proposed methodology for classifying swarm behavior with respect to computer network communications in Chapter VI.

4.1 Swarm Model Description

There are numerous characteristics that must be considered when developing a model for particle swarm behavior as seen in nature. These characteristics include model fidelity, particle behavior capabilities[1], and implementation method. The model fidelity can range

[1]These capabilities include but are not limited to speed, size, maneuverability, endurance, and sensor types.

over a spectrum that has one end in extreme simplicity using coarse linear models to the other end where every conceivable notion is modeled as closely as possible. For obvious reasons, a simplistic model is insufficient. However, approaching the other end of the spectrum is arduous at best and likely impossible since all of the interactions of natural swarm systems are not fully understood. The proposed model captures the local interactions and manifested global behavior while maintaining tractability. Tractability in this context is defined by simulation time. Ideally, simulations should run in real-time. However, due to computational complexity, this is not always the case.

It is assumed that a swarm consists of a number of homogeneous particles acting independently according to the *local* rules specified in Section 2.3.1. The algorithm described allows for non-homogeneous particle swarms, but for simplicity, only homogeneous swarms are considered.

The model was implemented in MATLAB^{TM} (MathWorks, Inc.) initially because of the ease and rapidity of development (65). However, it was soon evident that MATLAB was inefficient for a model of sufficient fidelity with a large number of nodes–on the order of 100–running for a reasonable simulation time–on the order of several thousand time steps. For this reason, the model was ported to Visual *C++*^{TM} (Microsoft Corp.). This provided the additional benefit of an effective visualization environment that was used to gain insight into swarm behavior (66). A detailed description of the simulator (functionality and usage) is provided in Section 4.2.

The focus of the remainder of this section is on the mathematical model. A new, innovative model is developed in order to provide a method for investigating behavior dynamics and relating those dynamics to the parameters that control swarm behavior. For simplicity, a two dimensional swarm model is considered. The model is made up of an algorithm for managing the time progression of the simulation as well as the formulas needed to compute particle movements. The significance of the model lies in both the formulas used as well as the system variables and parameters. The system variables are listed in Table 4.1. Additionally, the scope of each parameter is provided. The *local* parameters apply to each individual particle while the *global* parameters apply to the swarm as a whole. The parameters include numerous weighting factors. Values for the

Table 4.1 System Parameters

Parameter	Scope
Max speed	Local
Turning radius	Local
Separation dist	Local
Neighborhood size	Local
Population size	Global
Region size	Global

system parameters are chosen experimentally (i.e. values that make swarm formation movement stable–defined formally in Section 4.3–with respect to a desired behavior, see Appendix E). A goal of this work is to develop the relationships between the system parameters and the *continuum* of formation behavior.

The model is implemented using a discrete-time model. This is a simplifying assumption since a continuous-time model is more complex. For continuous-time implementations, complex formulas must be used to compute not only the updates but also the time steps for the updates.

The general simulation algorithm, shown in Figure 4.1, is described briefly as follows. For each *mobile* particle in the swarm, a new direction vector is calculated. This is done by considering *nearest neighbors*, *boundaries*, and *waypoints*. More precise definitions of each of these terms is given in Sections 4.1.3.1, 4.1.3.2, and 4.1.3.4 respectively.

```
Loop $\forall p_i \in S, i = 1, .., N$
   Process boundaries
   Loop $\forall p_j \in S_i, j = 1, ..., N_i$
      Process neighbor $p_j$
      Calculate new direction
   end Loop
   Move in new direction
end Loop
```

Figure 4.1 General Swarm Algorithm

The variables of Figure 4.1 are described in Table 4.2.

The ideal separation distance, d_{min}, is normalized to unity for all simulations. This allows for scaling to a desired separation for a specific application. It is also assumed

Table 4.2 Swarm Algorithm Variables

Variable	Description
S	The set of mobile particles
N	The population size (mobile particles), $\lvert S\rvert$
p_i	The i^{th} particle in S
S_i	The set of particles in p_i's *neighborhood* (includes waypoints)
N_i	The number of particles in p_i's neighborhood, $\lvert S_i\rvert$
p_j	The j^{th} particle in S_i

that swarm particles are point masses with physical dimensions that are negligible when compared to the separation distance.

4.1.1 Visibility Model. The set of neighbors for a particle consists of those particles that are *visible*. The concepts of *shadow*, *blocking*, and *visibility* is described by Definitions 4.1.1 through 4.1.3.

Definition 4.1.1 (Shadow) *A shadow of a particle is the set of all points $\boldsymbol{p}_k = (x, y)$ such that the following equations hold.*

$$vis(\theta_{ab}) = \begin{cases} true & : \theta_{ab} > \theta_{vis} \\ false & : \theta_{ab} \leq \theta_{vis} \end{cases} \tag{4.1}$$

$$\begin{aligned} \theta_{ab} &= \cos^{-1}\left(\frac{\boldsymbol{v}_a \cdot \boldsymbol{v}_b}{\|\boldsymbol{v}_a\| \cdot \|\boldsymbol{v}_b\|}\right) && (4.2) \\ \boldsymbol{v}_a &= \boldsymbol{p}_j - \boldsymbol{p}_i && (4.3) \\ \boldsymbol{v}_b &= \boldsymbol{p}_k - \boldsymbol{p}_j && (4.4) \end{aligned}$$

where θ_{ab} is the angle between vectors $\boldsymbol{v}_a$ and $\boldsymbol{v}_b$ and θ_{vis} specifies the amount of blocking (as illustrated in Figure 4.2).

Definition 4.1.2 (Blocking) *A particle is blocked if and only if it falls in the shadow of another particle.*

Definition 4.1.3 (Visibility) *A particle is visible by another particle if and only if it is not blocked.*

The *shadow* of a particle is defined with respect to the particle for which the neighborhood set is being populated. This is presented formally in Definition 4.1.1. It is important to note that the following description is based on a two dimensional model. The concept could be extended to a three dimensional model.

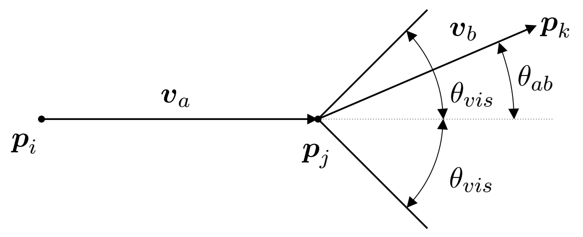


Figure 4.2 Visibility Model

In Figure 4.2 particle $\boldsymbol{p}_k$ is not visible by particle $\boldsymbol{p}_i$. A value of $\pi/3$ was chosen for θ_{vis}. The reason for this value is presented in Section 4.3. Basically, the maximum number of neighboring particles—to be proved—at the ideal separation distance is six. This divides the two dimensional region into six equal arcs of $\pi/3$. The visibility model serves to strongly restrict the size of a neighborhood set which is important for scalability. Additionally, it is assumed that a particle is visible regardless of the separation distance.

An equivalent form of Equation 4.1–based on trigonometric identities–is given in Equation 4.5. Note that the inequalities are reversed because of the nature of the cosine function. The angle θ_{ab} can take on values in the interval $[0, \pi]$; and, $\forall \theta_1, \theta_2 \in [0, \pi]$ if $\theta_1 \geq \theta_2$ then $\cos\theta_1 \leq \cos\theta_2$.

$$vis(\cos\theta_{ab}) \equiv \begin{cases} true & : \quad \cos\theta_{ab} < \cos\theta_{vis} \\ false & : \quad \cos\theta_{ab} \geq \cos\theta_{vis} \end{cases} \tag{4.5}$$

This form is important since the vector dot products can be computed and compared without the need for computing the inverse cosine.

Figure 4.3 provides a simple example of how the neighborhood set is determined. Particles $\boldsymbol{p}_b$ and $\boldsymbol{p}_d$ are blocked by particles $\boldsymbol{p}_a$ and $\boldsymbol{p}_c$, respectively. It is important to

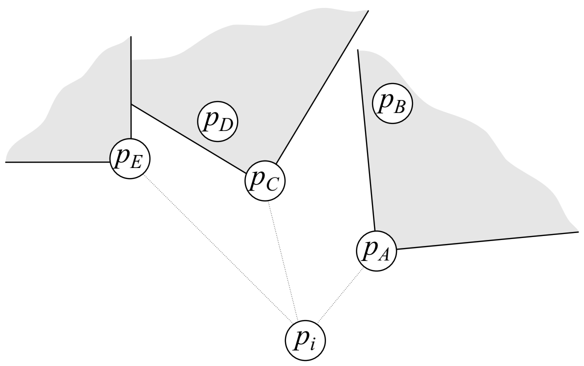


Figure 4.3 Visibility Example

Table 4.3 Particle State Elements

Variable	Type	Description
speed	scalar	The current speed
$turn_{max}$	scalar	The maximum allowable turn in degrees
dir	vector	The current direction (a unit vector)
pos	vector	The current position

note that, even though $\boldsymbol{p}_d$ is closer to $\boldsymbol{p}_i$ than $\boldsymbol{p}_e$, $\boldsymbol{p}_d$ is not considered a member of the neighborhood set of $\boldsymbol{p}_i$.

The use of a vision model that restricts the size of the neighborhood set ensures that the algorithm is scalable. In other words, swarms of different sizes but the same set of parameters have the same general behavior. This quality is described in greater detail in Section 4.3.

4.1.2 State information and assumptions. The state information that is encapsulated with each particle is given in Table 4.3. In keeping with the swarming methodology (99), these elements are maintained for each particle and are completely independent of other particles. Having said that, it should be noted that the $turn_{max}$ parameter is a fixed value and is the same for all particles. However, this allows for the capability to model a swarm of heterogeneous particles. The units of measurement and time are abstract entities for the purposes of the swarm simulation. For instance, *speed* is understood to be in distance units per one time unit. Further, it is assumed that the maximum speed is unity. This allows the swarm simulation to be adapted to any physical application with an appropriate transformation.

4.1.3 Swarm algorithm. This discussion provides the details of the swarming algorithm as performed by each swarm member independently including *neighborhood*, *peripheral vision*, and *vision blocking* models. The discussion also includes the use of boundaries and waypoints. Boundaries serve to limit the movement of the swarm to a specified rectangular region while waypoints are stationary particles that serve to guide the swarm along some predetermined route while allowing the swarm to maintain its properties. The process of updating the position of particle $\boldsymbol{p}_i$ generates a target vector that is updated with respect to each boundary and the particles (including waypoints) in the neighborhood of $\boldsymbol{p}_i$.

The desire is to capture the true functionality of the biological process that enables swarming to take place in nature as manifested by flocks of birds or herds of buffalo. That functionality is a continuous physical process that involves instantaneous feedback. In order to simplify the modeling process, the continuous system is approximated by a discrete-time model with feedback based on time steps. The discrete-time model lends itself well to a direct implementation for a computational model.

As described in Section 2.3.1, particle positions are updated according to three simple rules based on *separation*, *alignment*, and *cohesion*. *Separation* and *alignment* can be represented by the same vector quantity but with opposite magnitude. The position update vector $\boldsymbol{v}_{update}$ is computed for each particle, $\boldsymbol{p}_i$, in the swarm according to Equation 4.6. The *separation/cohesion* rules are encapsulated by the $\boldsymbol{v}_{attract}$ vector. The *alignment* rule is encapsulated by the $\boldsymbol{v}_{align}$ vector.

$$(\boldsymbol{v}_{update})_i = \sum_{p_j \in S_i} \left[(w_{periph})_{ij}(w_d)_{ij}\left((w_{attract})_{ij}(\boldsymbol{v}_{attract})_{ij} + C_{align}(\boldsymbol{v}_{align})_{ij}\right)\right] \tag{4.6}$$

The subscript notation of the indices i and j are used to clearly indicate that the various components of Equation 4.6 (with the exception of the constant C_{align}) are computed with respect to particles $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ where $\boldsymbol{p}_j$ is a member of the neighborhood set for $\boldsymbol{p}_i$. For the remainder of these discussions, the index subscripts are left off for the sake of brevity as shown in Equation 4.7.

$$(\boldsymbol{v}_{update})_i = \sum_{p_j \in S_i} \left[w_{periph} w_d \left(w_{attract} \boldsymbol{v}_{attract} + C_{align} \boldsymbol{v}_{align} \right) \right] \tag{4.7}$$

The particle direction vector $\boldsymbol{v}_{dir}$ and position are updated according to Equations 4.8 and 4.9, respectively.

$$\boldsymbol{v}'_{dir} = \frac{\boldsymbol{v}_{dir} + \boldsymbol{v}_{update}}{|\boldsymbol{v}_{dir} + \boldsymbol{v}_{update}|} \tag{4.8}$$

$$\boldsymbol{p}'_i = \boldsymbol{p}_i + (\boldsymbol{v}_{dir} + \boldsymbol{v}_{update})\Delta t \tag{4.9}$$

The quantity Δt is assumed to be unity. The exact implementation of Equation 4.9 is given by Equation 4.28 and described in detail in Section 4.1.3.6. The new particle direction and position are denoted by $\boldsymbol{v}'_{dir}$ and $\boldsymbol{p}'_i$ respectively. The sum in Equation 4.6 is accomplished over the set of neighbors S_i for particle $\boldsymbol{p}_i$. Each of the components in Equation 4.6 is described in detail in the following subsections. It is important to note that the direction vector always has unit magnitude while the position update is limited to a specified maximum distance movement–this ensures that particle movement is limited to a finite distance. This aspect of the model is explored in detail in Section 4.3.3.

The quantities $\boldsymbol{v}_{attract}$ and $\boldsymbol{v}_{align}$ are computed according to Equations 4.10 and 4.11, respectively.

$$\boldsymbol{v}_{attract} = \boldsymbol{p}_j - \boldsymbol{p}_i \tag{4.10}$$

$$\boldsymbol{v}_{align} = (\boldsymbol{v}_{dir})_j \tag{4.11}$$

where particle $\boldsymbol{p}_j$ is a neighbor of particle $\boldsymbol{p}_i$ (i.e., $\boldsymbol{p}_j \in S_i$). The quantity C_{align} is a constant and specifies the weighting of the *alignment* rule. The effect of this value on the behavior of the swarm is described in Section 4.3. The quantities w_d and w_{periph} are described in Sections 4.1.3.1 and 4.1.3.3 respectively. Briefly, the w_d parameter specifies the distance weighting, and the w_{periph} parameter specifies the peripheral vision weighting. It is important to note that the preceding description is a general definition. Particle $\boldsymbol{p}_j$ is typically another swarm member, but may also be a boundary or a waypoint.

The vector $\boldsymbol{v}_{attract}$ *points* from particle $\boldsymbol{p}_j$ to particle $\boldsymbol{p}_i$. Also, the quantity $(\boldsymbol{v}_{dir})_j$ is the direction of $\boldsymbol{p}_j$. An additional quantity not explicitly expressed is the distance between particles $\boldsymbol{p}_j$ and $\boldsymbol{p}_i$. The distance d is computed as

$$d \equiv \|\boldsymbol{v}_{attract}\|. \tag{4.12}$$

These symbols are used throughout the following discussions.

4.1.3.1 Neighborhood model. There are two models for the different types of "neighbors" that a particle might have. The first model is for boundaries and waypoints. The second is for other particles. The models partition the space around each particle into regions. The affect that another element (boundary, waypoint, or particle) has depends on which region the element is located in. Figure 4.4 shows the two neighborhood models.

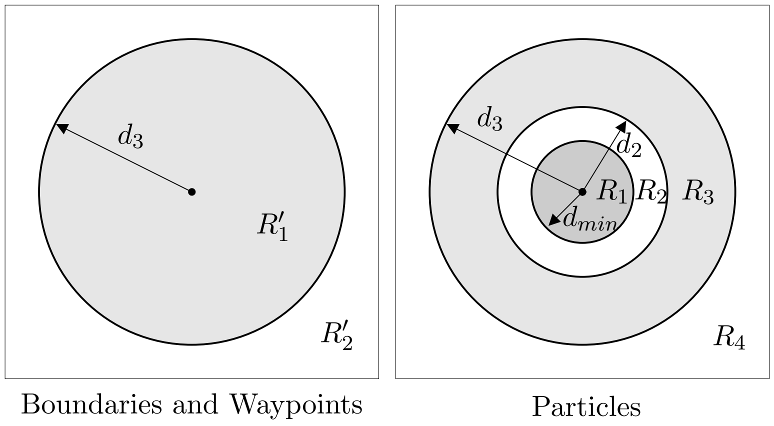


Figure 4.4 Neighborhood models

The rectangular regions represent all space. For boundaries and waypoints, this region is R'_2; for other particles, this region is R_4. There is only one other region, R'_1 for boundaries and waypoints. If the distance between a particle and a boundary or waypoint is d_3 then that boundary/waypoint is in region R'_1. For region R'_2 a particle is unaffected by the boundary/waypoint. For particles, the region is divided in a different manner when other particles are considered. Table 4.4 describes the meanings of each region. As stated, the value for d_{min} is unity. The value for d_2 is specified by a *comfort zone* parameter c_{zone}. As the value increases from zero, the size of region R_2 grows providing a larger region of

Table 4.4 Particle Neighborhood Regions

Region	Description
R_1	*Close*
R_2	*No effect*
R_3	*Far*
R_4	*Very far*

independence from neighbors. The intent is to provide greater flexibility to particles when computing a new direction. More details concerning the effect of this parameter on swarm behavior is given in Section 4.3. For particles, the difference between R_3 and R_4 manifests in the method for computing the effect of one particle on another as described.

The "neighborhood model" is used to compute the distance weight, w_d, of Equation 4.6 for each of the neighbor types. These types include boundaries, waypoints, and other swarm members. After determining the neighborhood set, the neighbors of $\boldsymbol{p}_i$ (including waypoints) are sorted into a list according to their distance from $\boldsymbol{p}_i$ with the closest particle first. The particles in this list are processed in this order so that the new direction for $\boldsymbol{p}_i$ can be properly determined.

4.1.3.2 Boundaries. The boundaries are processed by treating the closest point to $\boldsymbol{p}_i$ on each boundary as a particle that must be strongly avoided. Boundaries do not have direction so there is no $\boldsymbol{v}_{align}$ component. Equations 4.13 through 4.15 show how this is modeled.

$$\boldsymbol{v}_{attract} \equiv \boldsymbol{p}_b - \boldsymbol{p}_i,\ b \in \{north, south, east, west\} \tag{4.13}$$

$$w_d \equiv \begin{cases} 0 & : \quad p_b \in R_2' \quad (d < d_3) \\ \sqrt{d_3 - d} & : \quad p_b \in R_1' \quad (d \geq d_3) \end{cases} \tag{4.14}$$

$$w_{attract} \equiv \begin{cases} 0 & : \quad p_b \in R_2' \quad (d < d_3) \\ -C_{boundary} & : \quad p_b \in R_1' \quad (d \geq d_3) \end{cases} \tag{4.15}$$

The constant $C_{boundary}$ in Equation 4.15 is determined to be 30±5 by experimentation (see Appendix E). Smaller values allow particles to more closely approach the boundaries. This has the adverse affect of forcing the particles too close together. Larger values result in an unnecessary limitation of the region of movement. The vector $\boldsymbol{v}_{attract}$ is used for attraction and repulsion. The delineation is made by the sign—positive for attraction and negative for repulsion.

4.1.3.3 Peripheral vision. A model of peripheral vision is used to enhance the "fidelity" of the swarm algorithm. Objects that are *in front* of a particle are weighted more heavily than objects behind a particle. The formula for computing the peripheral weighting is given in Equation 4.16. Plots of w_{periph} for various values of n are shown in Figure 4.5.

$$w_{periph}(\theta) \equiv C_{periph} \cos^n \left(\frac{\theta}{2}\right), \quad n = 1, 2, 3, ... \tag{4.16}$$

The value used for C_{periph} was chosen to be unity since other factors–see Equation 4.6–scale this weight. Values of $n < 2$ ensure that no neighboring particles are completely

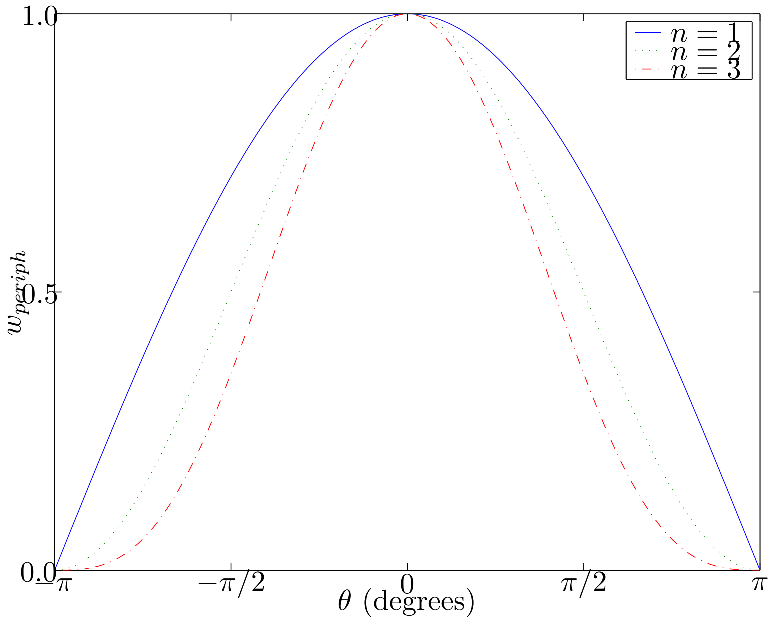


Figure 4.5 Peripheral Weight

"ignored" when computing the update vector for each particle. This is due to the fact that the values of $w_{periph}(\theta)$ are not zero near $\pm\pi$ except at $\theta = \pi$. It is also important to note that Equation 4.16 can be rewritten, through the trigonometric identity

$$w_{periph}(\theta) = C_{periph}\left[\frac{1}{2}(1+\cos\theta)\right]^{\frac{n}{2}}. \tag{4.17}$$

The quantity $\cos\theta$ can be computed from the dot product of the vectors shown in Figure 4.6. The direction of particle $\boldsymbol{p}_i$ is given by $\boldsymbol{v}_{dir}$ and the vector $\boldsymbol{v}_{attract}$ is computed

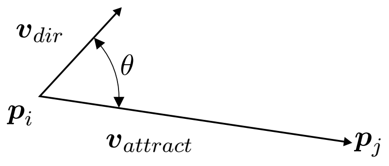
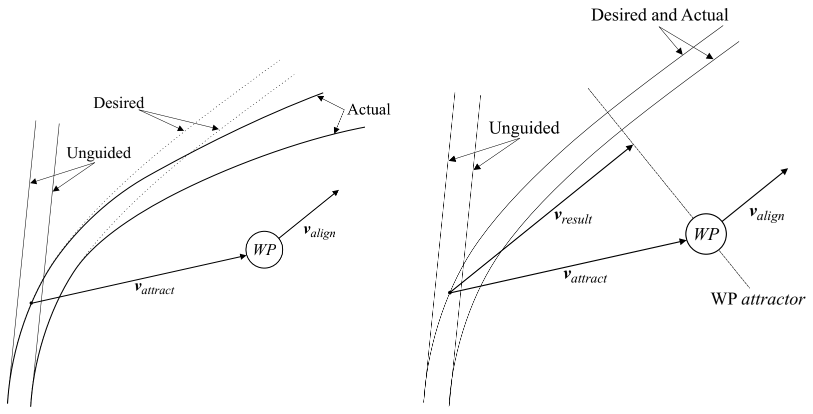


Figure 4.6 Peripheral Weight

according to Equation 4.10. Since $\boldsymbol{v}_{dir}$ is a unit vector the quantity $\cos\theta$ can be computed using the vector dot product

$$\cos\theta = \boldsymbol{v}_{dir}\cdot\frac{\boldsymbol{v}_{attract}}{|\boldsymbol{v}_{attract}|}. \tag{4.18}$$

4.1.3.4 Waypoints. Waypoints are fixed particles with direction along the intended route. A waypoint is defined as a navigation fix–usually a destination or point of reference. It is important that waypoints not affect the swarming behavior with respect to separation, cohesion, and alignment, but they should guide the swarm along an intended route. Ideally, swarm behavior should be consistent for a given set of parameters regardless of whether waypoints are present or not–local behavior should not be affected and the only global behavior effect should be the direction of the swarm. The use of waypoints is illustrated in Figure 4.7. In Figure 4.7 (a) the particles are attracted to the waypoint $\boldsymbol{p}_{wp}$ itself. The vector $\boldsymbol{v}_{attract}$ in (a) demonstrates this situation. In (b) the particles are attracted to the closest point on a line that passes through the waypoint and is perpendicular to the waypoint direction $\boldsymbol{v}_{align}$ (this is the *dir* element described in Table 4.3). This is accomplished by computing the component of $\boldsymbol{v}_{attract}$ in the direction of $\boldsymbol{v}_{align}$. The formula for this is given in Equation 4.19. The dot product $\boldsymbol{v}_{attract}\cdot\boldsymbol{v}_{align}$ computes the magnitude. Since $\boldsymbol{v}_{align}$ is a unit vector, the result $\boldsymbol{v}_{result}$ is a vector in the direction of $\boldsymbol{v}_{align}$ with the magnitude equal to the component of $\boldsymbol{v}_{attract}$ in the direction



(a) (b)

Figure 4.7 Swarm Behavior

of $\boldsymbol{v}_{align}$.

$$\boldsymbol{v}_{result} = \boldsymbol{v}_{align}(\boldsymbol{v}_{attract} \cdot \boldsymbol{v}_{align}) \tag{4.19}$$

It is the $\boldsymbol{v}_{result}$ vector that is used in Equation 4.21 to calculate the distance weight for $\boldsymbol{p}_{wp}$. Further, the distance d in Equations 4.20 and 4.21 is the norm of $\boldsymbol{v}_{result}$.

$$w_d \equiv \begin{cases} \left(\frac{d}{d_3}\right)^2 \left(-\boldsymbol{v}_{align} \cdot \frac{\boldsymbol{v}_{result}}{d}\right) & : \quad \boldsymbol{p}_{wp} \in R_1' \quad (d < d_3) \\ \left(e^{\frac{-(d-d_3)}{d_3}}\right) \left(-\boldsymbol{v}_{align} \cdot \frac{\boldsymbol{v}_{result}}{d}\right) & : \quad \boldsymbol{p}_{wp} \in R_2' \quad (d \geq d_3) \end{cases} \tag{4.20}$$

$$w_{attract} \equiv \begin{cases} -C_{wp} & : \quad \boldsymbol{p}_{wp} \in R_1' \quad (d < d_3) \\ C_{wp} & : \quad \boldsymbol{p}_{wp} \in R_2' \quad (d \geq d_3) \end{cases} \tag{4.21}$$

The vector dot product term provides an attraction/repulsion mechanism so that particles *behind* the waypoint are attracted while particles in front are repulsed (regardless of particle direction). This term varies in the range $[-1, 1]$ since $\boldsymbol{v}_{align}$ and $\frac{\boldsymbol{v}_{result}}{d}$ are unit vectors. The constant C_{wp} in Equation 4.21 is determined to be 10 ± 5 through experimentation. The $\boldsymbol{v}_{align}$ component of Equation 4.6 is simply the direction $(\boldsymbol{v}_{dir})_{wp}$ of the waypoint.

4.1.3.5 Swarm members. If a neighboring particle $\boldsymbol{p}_j$ is not a waypoint then the calculations for attraction/repulsion are made according to Equations 4.22 and 4.23. For $\boldsymbol{p}_j \in R_1$ the effect is repulsion and for $\boldsymbol{p}_j \in R_3, R_4$ the effect is attraction (as indicated by the sign in Equation 4.23).

$$w_d \equiv \begin{cases} \sqrt{1-d} & : \; \boldsymbol{p}_j \in R_1 \quad (d < d_{min}) \\ 0 & : \; \boldsymbol{p}_j \in R_2 \quad (d_{min} \le d < d_2) \\ \left(\frac{d-d_2}{d_3-d_2}\right)^2 & : \; \boldsymbol{p}_j \in R_3 \quad (d_2 \le d < d_3) \\ e^{\frac{-(d-d_3)}{d_3}} & : \; \boldsymbol{p}_j \in R_4 \quad (d \ge d_3) \end{cases} \tag{4.22}$$

$$w_{attract} \equiv \begin{cases} -C_{repulse} & : \; \boldsymbol{p}_j \in R_1 \quad (d < d_{min}) \\ 0 & : \; \boldsymbol{p}_j \in R_2 \quad (d_{min} \le d < d_2) \\ C_{attract} & : \; \boldsymbol{p}_j \in R_3 \quad (d_2 \le d < d_3) \\ C_{attract} & : \; \boldsymbol{p}_j \in R_4 \quad (d \ge d_3) \end{cases} \tag{4.23}$$

The values used for the weights $C_{repulse}$ and $C_{attract}$ are described in Section 4.3. The values for these parameters can be varied in order to produce different global behaviors such as *coherent behavior* or *incoherent behavior* (see Table 4.7).

A plot of w_d from Equation 4.22 is shown in Figure 4.8. The horizontal axis is the distance between particles. The labelled regions R_1 through R_4 correspond to the regions described in Figure 4.4. As stated, the separation distance is normalized so that d_{min} is

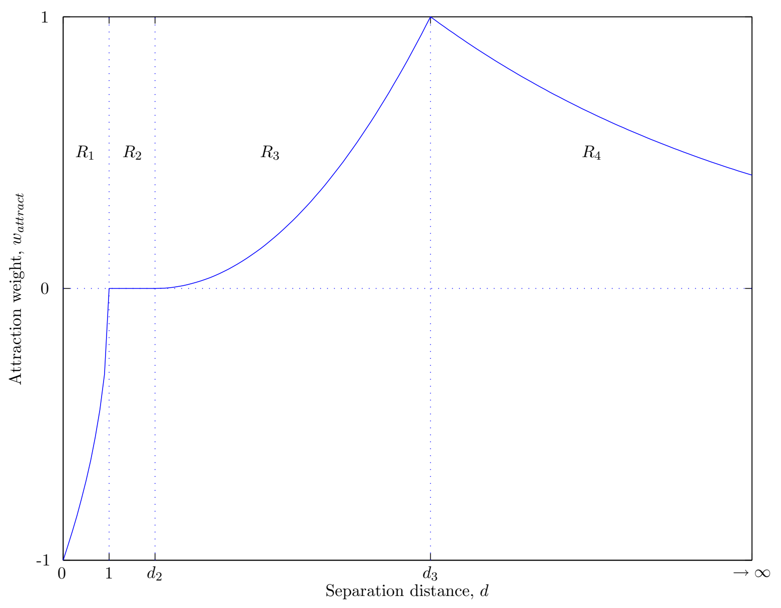


Figure 4.8 Particle Attraction Weight

unity. The values for the transitions from R_2 to R_3 and R_3 to R_4 are given by Equations

4.24 and 4.25 respectively.

$$d_2 \equiv 1 + c_{zone} \tag{4.24}$$

$$d_3 \equiv R_{max} d_2 \tag{4.25}$$

The comfort zone parameter c_{zone} is defined in Section 4.1.3.1. The R_{max} parameter sets the radius at which the attraction weight stops increasing and begins to decrease (see Figure 4.8). The value for R_{max} in Equation 4.25 is determined to be 4 ± 2. Too small a value results in weak cohesion while too large a value results in a tight formation that tends to frequently violate the separation rule. The important point to note about Figure 4.8, as pointed out in (32), is that, at $d = 1$, particles are in a state of equilibrium—i.e. the weight is zero and particle p_j has no effect on p_i. However, this premise holds only for three or less particles since the addition of one or more particles must result in at least one particle with a non-zero weight[2]. Algorithms that fail to account for larger swarms result in non-scalable algorithms. This is discussed in greater detail in Section 4.3.

4.1.3.6 Movement. As stated earlier, the direction updates are combined as shown in Equation 4.6. The update vector specifies both the direction and the magnitude for the movement of particle $\boldsymbol{p}_i$. However, in keeping with realistic vehicle models, a particle is restricted in both the magnitude and direction updates. The magnitude limit is achieved by ensuring that a maximum velocity is not exceeded through the proper choice of swarm parameters. A detailed discussion of this situation is given in the next section.

The direction limit is achieved by enforcing a hard limit on the turn amount. The amount a particle can turn is limited by the *maximum turn angle* (see Table 4.3), θ_{max}. Therefore, the new direction of particle $\boldsymbol{p}_i$ is given by the minimum of θ_{max} and θ_{update}. This is illustrated for a *left* turn in Figure 4.9 where the current particle direction is given by θ_i. Since $\Delta\theta > \theta_{max}$ the new direction for particle $\boldsymbol{p}_i$ is updated to $\theta_i + \theta_{max}$ where θ_i is the current direction. The value for θ_{max} is set globally for the swarm population. The current value is set at 4^o. Larger values can be used but require a lower limit on particle velocity. In order to provide a random element to the update process, the update vector

[2]This assumes no form of vision blocking is used

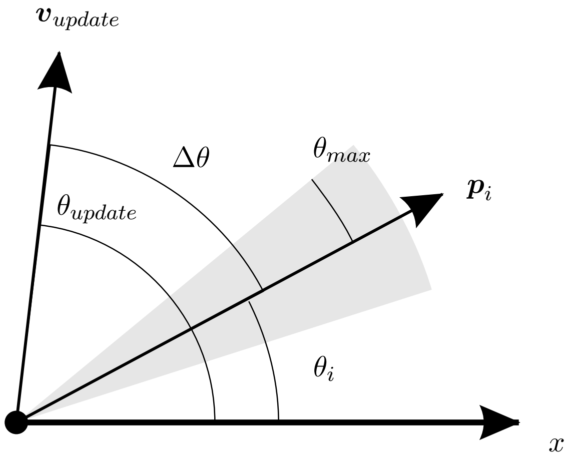


Figure 4.9 Particle Turn Example

direction is perturbed by a value θ_p chosen from a uniform distribution in the interval $[-\theta_{max}, \theta_{max}]$. For scenarios explored in this investigation, θ_{max} was chosen to be 4^o. The purpose of the perturbation is to provide a stochastic element that allows investigation into the steadiness of the swarming algorithm in the presence of system noise. The new *change* in direction ${\theta'}_{update}$ is given by Equation 4.26

$$ {\theta'}_{update} = \mathtt{sgn}(\theta_{update} + \theta_p)\mathtt{min}(\theta_{max},\ |\theta_{update} + \theta_p|) \tag{4.26} $$

where $\mathtt{sgn}()$ computes the sign of its argument and $\mathtt{min}()$ computes the minimum of its two arguments. The new direction ${\theta'}_i$ for particle $\boldsymbol{p}_i$ is given by Equation 4.27.

$$ {\theta'}_i = \|\theta_i + {\theta'}_{update}\|_{360} \tag{4.27} $$

where the $\|\cdot\|_{360}$ operator is a modulus function about 360^o.

The update vector is scaled by N_i, the neighborhood size of particle $\boldsymbol{p}_i$. This ensures that the update vector is independent of neighborhood size. This is derived and described in detail in the next section. The new particle position $\boldsymbol{p}'_i$ is given by Equation 4.28.

$$ \boldsymbol{p}'_i = \boldsymbol{p}_i + \alpha_s\left(\boldsymbol{v}_{dir} + \frac{\boldsymbol{v}'_{update}}{N_i}\right) \tag{4.28} $$

where $\boldsymbol{v}'_{update}$ has the same magnitude as $\boldsymbol{v}_{update}$ in Equation 4.6 but the direction has been adjusted to θ'_i so as not to exceed θ_{max} (Equations 4.26 and 4.27). The parameter α_s scales the position update so that the particle movement is "closed" with respect to particle movement capabilities[3]. Additionally, the vector $\boldsymbol{v}'_{update}$ is normalized with respect to the neighborhood size N_i. The vector $\boldsymbol{v}_{dir}$ is the current direction of the particle and has unit magnitude.

4.2 Simulator Implementation

As stated in Section 4.1 a Microsoft Windows-based multi-threaded simulator was developed in Visual C++. This section documents the program structure and the mapping of logical elements to programmatic implementation. Usage of the graphical and command-line versions for generating swarm track visualizations and swarm track histories is described in Appendix F.

The program provides a mechanism to evaluate swarm behavior by visualizing the simulation as it progresses. Figure 4.10 shows an example swarm simulation. The tracks

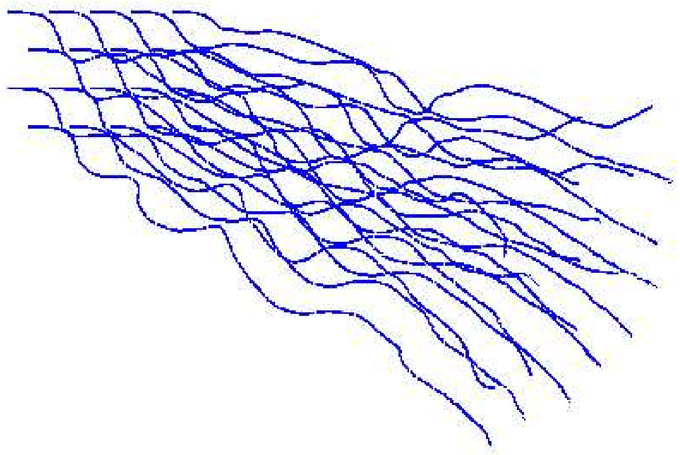


Figure 4.10 Sample Swarm Simulation

represent the particles movement over time. The swarm of 20 particles is initialized in a

[3]It is assumed that particles have limited movement capabilities. These limits affect acceleration, velocity, and turning radius.

lattice formation in the upper left of the figure and moves toward the lower right. While the individual tracks are difficult to make out in Figure 4.10, the use of the simulator provides an animation capability that makes it easy to see the swarm behavior.

The simulator plots the particles as red points and the tracks in blue. At regular intervals, the simulator plots the logical connections between particles as green edges (lines). A connection exists if two particles are less than $(1 + c_{zone})$ distance units apart. If the separation is less than unity, the edge is red. Additionally, whenever the simulation is stopped, the formation connections are painted.

4.2.1 Program Structure. This section presents the object-oriented data structures that make up the computational elements of the swarm simulator. The following objects are explained in this section. There are numerous references for an explanation of the Microsoft Visual *C++* objects (24, 36, 63, 81).

- `CFormation`
- `CParticle`
- `CParams`
- `CStep`
- `CMatrix`
- `CVector`

By convention, class names begin with an uppercase "`C`" and member variable names begin with a lowercase "`m_`" (80). The swarm classes are presented and discussed in the reverse order. For each class, a description is provided for the constructors/destructors, member variables, and member functions. The variables and functions (including constructors/destructors) are presented as they appear in the declaration header files.

```
CVector();
CVector(CVector &other);
CVector(double x_new, double y_new);
CVector(double dir);
CVector(CPoint pt);
virtual ~CVector();
```

Figure 4.11 Vector Constructors/Destructors

printPolr	[0.0000 / 0.0000]
printRect	[0.0000 0.0000]

4.2.2 CVector. The CVector and CMatrix classes make up the fundamental objects of the swarm simulator. The CVector class is used to store particle position and velocity information. The constructors shown in Figure 4.11 generate a new CVector instantiation from another CVector object, from a location (or velocity), and from a direction. The direction is given in degrees and is measured counterclockwise from the positive x-axis. The result of the last constructor is a unit vector in the specified direction. The CVector destructor has no functionality. The member variables are simply the x and y components. These values are stored as double precision floating point variables.

```
double toAngle(double perturb = 0);
void makeUnitVec();
void fromAngle(double theta);
double norm();
CVector &operator=(CVector &other);
CVector operator+(CVector &other);
CVector &operator+=(CVector &other);
CVector &operator*=(double scalar);
CVector operator*(double scalar);
double operator*(CVector &other);
CVector operator/(double scalar);
CVector operator-(CVector &other);
void printPolr(char *buff);
void printRect(char *buff);
```

Figure 4.12 Vector Member Functions

The CVector member functions, shown in Figure 4.12, provide algebraic support for manipulating vectors with the exception of the latter two functions. The algebraic support includes multiplication and division by a scalar, vector dot product, as well as vector addition and subtraction. The last two functions provide formatted I/O. The output format of these functions is shown below.

For printPolr, the first value is the magnitude and the second value is the angle measured counterclockwise from the positive x-axis. The printRect format is simply the x and y values.

4.2.3 CMatrix. The `CMatrix` class is used to maintain information regarding inter-particle distance as well as several specialized pieces of information as described below. The constructor is shown below.

```
CMatrix(int rows, int cols);
```

It initializes the matrix with sufficient storage for a double precision floating point matrix that is `rows` by `cols`. The member variables include two integers for specifying the number of rows and columns (`m_Rows` and `m_Cols` respectively) as well as a pointer `m_pData` to the double precision array for storing the data. Various methods, shown in Figure 4.13, are

```
double ColMin(int col);
double GetAt(int row, int col);
void SetAt(int row, int col, double val);
```

Figure 4.13 Matrix Member Functions

used to manipulate the matrix data. It should be noted that the `CMatrix` class is generic in its declaration so that it may be reused without modification in other applications.

4.2.4 CStep. The `CStep` class is used to store time step data. At each time step of the simulation, the swarm formation data is stored in a list (see `m_listMoves` in Section 4.2.7. This list stores swarm history information that can be subsequently stored to a binary file (see Section F.1.1). The constructors, shown in Figure 4.14, are used to initialize the particle array as well as providing for object replication.

```
CStep(int size);
CStep(CStep &other);
CStep();
```

Figure 4.14 Step Constructors/Destructors

The `CStep` class has one member variable and one member function. These are shown below.

```
CArray<CParticle, CParticle> m_pointArray;

CStep &operator=(CStep &other);
```

The `m_pointArray` is used to store formation information at each time step.

4.2.5 CParams. The `CParams` class maintains global state information (parameters) of the swarm. These parameters specify the global settings for the swarm formation. In addition to the standard constructor/destructor a constructor is used to enable object copying. This constructor is shown below.

```
CParams(CParams &other);
```

The state data includes the number of swarm members and waypoints (`m_Popsize` and `m_Waypoints` respectively). The `m_ptrX` pointer variable is used to store dynamic weight parameters (see Section F.1.1, `Set Param File...`). The `m_iTime` variable is used to count the number of time steps in the simulation. The other parameters are used when the swarm is initialized. The `CParam` member variables are listed in Figure 4.15.

```
BOOL m_bBounded, m_bDynParams, m_bShowDir;
void *m_ptrX;
int m_iTime;
float m_MaxTurn;
float m_Velocity;
double m_CZone;
int m_RSeed;
float m_MoveDir;
int m_PopSize, m_Waypoints;
```

Figure 4.15 Param Member Variables

The `CParam` class has only one member function. The assignment operator is shown below.

```
CParams &operator=(CParams &other);
```

4.2.6 CParticle. The `CParticle` class is the structure used to encapsulate a single particle and is used for both swarm members and waypoints. In addition to the standard constructor/destructor, two other constructors are used. One is for simple object replication while the other is used to create and initialize a particle object. These are listed in Figure 4.16.

Figure 4.17 shows the `CParticle` member variables. The `m_bWP` variable serves as a flag to indicate whether the instantiation is for a swarm member (the value is `FALSE`) or a waypoint (the value is `TRUE`). The `m_Region` variable defines the boundaries for the

```
CParticle(CParticle &other);
CParticle(CVector init, double dir);
CParticle();
virtual ~CParticle();
```

Figure 4.16 Particle Constructors/Destructors

swarm simulation if it is enabled (see Sections F.1.1 and F.2). The `m_Speed`, `m_maxSpeed`, and `m_maxTurn` are currently held constant at values specified by the `CParams` object. It is envisioned that the simulator would support swarms of non-homogeneous particles. In which case, these parameters would be varied for each particle. The individual particle state information is maintained by the `m_Dir` and `m_Pos` variables. The `m_Dir` variable is a unit vector in the direction of movement. The `m_Pos` variable specifies the (x, y) location. The remaining variables are used to track diagnostic data for each simulation. These

```
BOOL m_bWP;
CSize m_Region;
double m_Speed, m_maxSpeed;
double m_maxTurn;
double m_Dir;
CVector m_Pos;
int m_setSize;
double m_dirM, m_distM, m_distMin, m_distMax;
```

Figure 4.17 Particle Member Variables

measures are computed for each particle at each time step. The data is based on the set *visible* neighbors for each particle. The `m_setSize` variable is used to capture the size of the visible set, $N_i = |S_i|$ (see Section 4.1 and Table 4.2). These measures are discussed in detail in Section 6.1. They are reviewed briefly here. The methods with which they are computed are captured in Equations 4.29 through 4.33.

$$\begin{aligned}
\texttt{m\_setSize}_i &= |S_i| && (4.29)\\
\texttt{m\_dirM}_i &= \sqrt{\frac{1}{N_i}\sum_{j\in S_i}(\theta_i-\theta_j)^2} && (4.30)\\
\texttt{m\_distM}_i &= \sqrt{\frac{1}{N_i}\sum_{j\in S_i}(d_j-1)^2} && (4.31)\\
\texttt{m\_distMin}_i &= \min_{j\in S_i}(d_j) && (4.32)\\
\texttt{m\_distMax}_i &= \max_{j\in S_i}(d_j) && (4.33)
\end{aligned}$$

As Section 6.1 states, these values are *distributed* measures that can be computed by each particle independently.

The member functions for the `CParticle` class are shown in Figure 4.18. The assignment operator is self-explanatory. The `setMaxTurn` function sets θ_{max} (see Section 4.1.3.6) for each particle. The `move` method computes a new position from a vector (not necessarily normalized). It implements the computation of Equation 4.28. The other parameters of this method include the speed factor, α_s (see also Section 4.1.3.6), a perturbation amount, and θ_{max} (see Figure 4.9). The `getPos` and `getDir` methods return the current location and

```
CParticle &operator=(CParticle &other);
void setMaxTurn(double t);
void move(CVector vecNew, double sfactor, double perturb, double tmax);
CVector getPos();
double getDir();
```

Figure 4.18 Formation Member Functions

direction of a particle as as a `CVector` object and a double precision number respectively.

4.2.7 CFormation. The `CFormation` class encompasses the complete encapsulation of the swarm. It maintains an array of particles and waypoints (if used) as well as the list of swarm movement history. The specialized constructor is shown below.

```
CFormation(CParams tParams);
```

A formation is constructed with a `CParams` object that specifies the global parameter settings as well as an optional dynamic parameter specification.

The member variables are listed in Figure 4.19 and the methods are listed in Figure 4.21. The `ParticleArray` data type is declared as follows.

```
typedef CArray<CParticle, CParticle> ParticleArray;
```

The variables `m_piNindx`, `m_pdNdist`, and `m_pdNperiph` are used by the `findNeighbors` method. These variables correspond to the symbols p_j, d, and w_{periph} in Table 4.2, and Equations 4.12 and 4.16 respectively. These variables are one-dimensional arrays that maintain a list (sorted by distance, nearest to farthest) of neighbors for a particular particle. For example, the particles around $\boldsymbol{p}_i$ in Figure 4.3 would be sorted as follows: $\boldsymbol{p}_a$, $\boldsymbol{p}_c$, $\boldsymbol{p}_e$, $\boldsymbol{p}_d$, and $\boldsymbol{p}_b$. Note that this ordering is obtained before the visibility model is used to limit the neighborhood for particle $\boldsymbol{p}_i$. Therefore, the `findNeighbors` method considers the entire swarm population when computing the values. The `m_piNset` variable is used as a flag to

```
int *m_piNindx;
double *m_pdNdist;
double *m_pdNperiph;
int *m_piNset;
struct param_rec *m_paramList;
static CTypedPtrList<CObList, CStep*> m_listMoves;
CParams m_Params;
CMatrix m_Pinfo;
ParticleArray m_arrayParticles;
ParticleArray m_arrayWPs;
```

Figure 4.19 Formation Member Variables

determine what particles are in the actual neighborhood (i.e., visible) of a particle. This corresponds to S_i in Table 4.2.

The `m_paramList` variable is a linked list of dynamic parameters. The structure of this element is shown in Figure 4.20. The relationship between this structure and the algorithmic symbols used in Section 4.1 is shown in Table 4.5.

As described above, the `m_listMoves` variable stores a history of the swarm movement in a linked list. The `m_Params` variable stores the global swarm parameters. The `m_Pinfo` variable is an N by N matrix that stores the particle distance data. It is updated after each particle movement. Finally, the particles and waypoints are maintained in the `m_arrayParticles` and `m_arrayWPs` arrays respectively.

```
struct param_rec {
  double t;
  double A, B, C, D, E, G;
  double vel_max;
  double turn_max;
  double czone;
  double p_max;
  struct param_rec *next;
};
```

Figure 4.20 Parameter Record Structure

Table 4.5 Dynamic Swarm Variables

Variable	Symbol	Desciption
`t`	N/A	Paramter set start time
`A`	$C_{boundary}$	Boundary scale factor
`B`	C_{periph}	Peripheral vision scale factor
`C`	C_{wp}	Waypoint scale factor
`D`	$C_{repulse}$	Repulsion factor
`E`	C_{align}	Alignment factor
`G`	$C_{attract}$	Attraction factor
`v_fac`	α_s	Speed scale factor
`turn_max`	θ_{max}	Maximum allowable turn in degrees
`czone`	c_{zone}	Comfort zone factor
`p_max`	θ_p	Perturbation factor

The `CFormation` member Functions are shown in Figure 4.21. The `init` method initializes the swarm formation by generating either a randomly placed swarm (`bRandom` set to `TRUE`) or a lattice formation. It also provides the initialization control for dynamic parameter specification. This method calls the `InitPinfo` method to initialize the `m_Pinfo` variable and compute the distances between all the particles.

The `findist` and `findistEx` methods compute the distances between a particle and the rest of the swarm members or all the particles (if a row value is not specified). The `findist` method performs the computations using the member variables `m_arrayParticles` and `m_arrayWPs` while the `findistEx` method performs the computations based on the specified particle array.

The collection of `set...` methods are used by the `init` method to set the maximum turn angle (`setMaxTurn`), particle speed (`setSpeed`), position (`setPos`), and direction

```
void init(BOOL bRandom = TRUE, char chrFmnfile[] = NULL);
void InitPinfo();
void findist(int row = -1);
void findistEx(CArray<CParticle, CParticle> &tarray,
  CMatrix *distInfo, int row = -1);
void setMaxTurn(int i, double t);
void setSpeed(int i, double s);
void setPos(int i, double x, double y);
void setDir(int i, double dir);
int getPopSize();
double getDir(int i);
void move_update();
void findNeighbors(int i, int nhood[], double ndist[], double nperiph[]);
double getPinfo(int i, int j);
void setDynParams(char filename[80]);
```

Figure 4.21 Formation Member Functions

(`setDir`). Each of these methods sets these values for a single particle in the formation (specified by the input argument `i`).

The `getPopSize` method return the size of the swarm. The `getDir` method returns the direction (in degrees, counterclockwise from the positive x-axis) for the particle with index `i`.

The `move_update` method implements the majority of the swarm simulation algorithm (as described in Chapter IV). As stated earlier, the `findNeighbors` method computes "neighborhood" information for the particle specified by the index `i`.

The `getPinfo` method returns the distance between the particles specified by the indexes `i` and `j`.

Finally, the `setDynParams` method provides support for using dynamic parameters. It opens the specified text file and reads the formatted `dyn` file. The format of this file is given in Section F.1.1.

4.3 Swarm Parameters

The set of parameters for the swarm model, formally defined by Equation 4.34 and listed in Table 4.6, determine the global behavior of the swarm formation. The effect of these parameters and a method for evaluating swarm behavior is described in this and the next section.

Some preliminary definitions and theorems must be stated to facilitate analysis of the swarm model parameters as they apply to swarm formation. The following notation is used to describe the various parameters of the algorithm: The set of parameters $\mathcal{P}$ is formally defined to be the set of all constants used in the swarming algorithm. This is shown in Equation 4.34

$$\mathcal{P} = \{C_{periph}, C_{align}, C_{repulse}, C_{attract}, c_{zone}, \alpha_s, \theta_p\} \tag{4.34}$$

There are other parameters $(C_{boundary}, C_{wp})$ that relate the swarm to the *environment*. These are not considered in this analysis since boundaries and waypoints are not used. It was determined, as an initial investigation of swarm behavior with respect to the system parameters, that only the swarm would be considered.

Table 4.6 Parameter Set Members

Param.	Description
C_{periph}	Peripheral vision
C_{align}	Alignment rule
$C_{repulse}$	Repulsion rule
$C_{attract}$	Attraction rule
c_{zone}	Comfort zone radius
α_s	Speed scale factor
θ_p	Update direction perturbation

There are global state specifications which are those *limits* imposed by physical implementation. These include maximum speed (normalized to unity), sight distance (R_{max}, Section 4.1.3.5), maximum turn angle (θ_{max}, Section 4.1.3.6), etc. The values for these specifications were held constant.

The values for the parameters in $\mathcal{P}$ are held constant in order to obtain a desired global swarm behavior–such as *flocking* or *schooling.* However, as stated in (22), the parameters of a system can vary because of other influences. The same is true for the generic swarm model[4]. This section is an exploration of the effect of parameter variations as it relates to swarm behavior.

4.3.1 Steadiness. Definition 4.3.1 provides a means of measuring a particular formation in terms of steadiness. For this and all subsequent definitions and theorems it is assumed that the parameters $\mathcal{P}$ in Equation 4.34 are always non-zero with the exception of c_{zone} which is assumed to be zero.

Definition 4.3.1 (Steadiness) *A steady swarm formation is one for which all update vectors are zero.*

$$(\boldsymbol{v}'_{update})_i = \mathbf{0} \quad \forall p_i \in S \tag{4.35}$$

Consider the update vector $\boldsymbol{v}'_{update}$ of Equation 4.28 above. If $\boldsymbol{v}'_{update} = \mathbf{0}$ then there is no change in the state of particle $\boldsymbol{p}_i$. In this fashion, $\boldsymbol{v}'_{update}$ is a measure of the *error* in a swarm formation at any given instant. If the perturbation factor θ_p is assumed to be zero and there are no other external influences, then the swarm simulation, as a purely deterministic algorithm, must maintain zero magnitude update vectors. This is the meaning of steadiness. The swarm is not changing within itself. The only external change is that of *translation*. A more relaxed definition would encompass *rotation* as well (since the relative positions within the swarm would not change).

Definition 4.3.1 is restrictive since it requires that c_{zone} be zero and does not address particle movement. In the presence of movement, the swarm can remain steady only if all the velocities are precisely equal since this results in no relative position change (and consequently, no change to the update vectors). This is not a realistic property. A relaxation of these requirements is discussed in the next section so that different behaviors may be related to steadiness.

Theorem 4.3.1 (Independence) *Swarm steadiness is independent of the parameters (both global and dynamic)*

[4]The system parameters may change, due to external, environmental influences, so that the behavior undergoes a form of bifurcation. This results in a completely different behavior.

Proof Steadiness is defined strictly in terms of the update vectors. Therefore, no other swarming aspects need be considered. The update vectors are algebraic functions of the parameters and the relative positions of the particles in the swarm. The update vectors are zero if all the parameters are zero or if all the weights are zero. Since by assumption, the parameters are non-zero, the weights must be zero. Therefore, the update vectors are zero independently of the parameters. ▮

The following theorem is presented as a step toward analyzing swarm formations in a formal sense

Theorem 4.3.2 *A triangular lattice structure is a steady formation.*

Proof By definition, the update vectors for a steady formation are zero magnitude. Then, it is sufficient to show that the triangular lattice has collective zero update vectors.

Consider a regular lattice structure (an example of such structure is shown in Figure 4.22). The vertices represent particles and the edges represent unit distance.

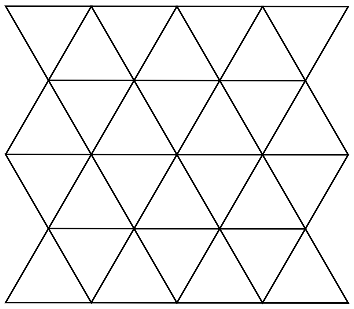


Figure 4.22 Sample Lattice Structure

Clearly the nearest neighbors make zero contribution to the update vector of each particle since each nearest neighbor particle is separated by one distance unit. However, the contributions by particles that are beyond the unit radius must be accounted for also.

Consider the simple lattice structure of the hexagon shown in Figure 4.23. Again, edges between particles indicate that they are separated by one distance unit. The hypothesis is that the ring of six particles around the center particle $\boldsymbol{p}_i$ completely block the view of any other particle outside of the radius r indicated in the figure. This radius represents

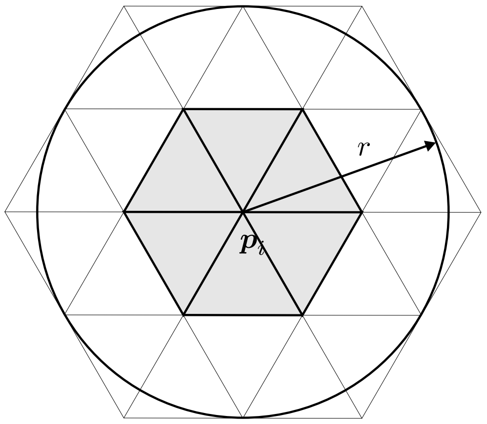


Figure 4.23 Simple Hexagonal Structure

the closest distance that any particle can be located without violating the requirement that all update vectors have magnitude zero. Because the triangles that make up the lattice are equilateral with unit sides, the value of r is $\sqrt{3}$.

The hexagon of Figure 4.23 naturally divides the space around $\boldsymbol{p}_i$ into six sectors with $\pi/3$ radians each. Without loss of generality, consider a single sector as shown if Figure 4.24. The particle $\boldsymbol{p}_k$ must be in the upper half of the sector ($0 \leq \theta_k \leq \frac{\pi}{6}$) or in the

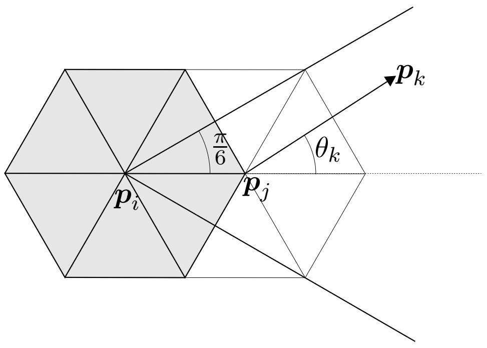


Figure 4.24 Single Sector

lower half ($-\frac{\pi}{6} \leq \theta_k \leq 0$). Because of symmetry, only the upper sector is considered. As stated in Definition 4.1.3, the particle $\boldsymbol{p}_k$ is visible if and only if Equations 4.1 through 4.4 hold.

So θ_k is given by

$$\theta_k = \frac{\boldsymbol{v}_a \cdot \boldsymbol{v}_b}{|\boldsymbol{v}_a||\boldsymbol{v}_b|} \tag{4.36}$$

By Equation 4.5 and the fact that $\cos(\pi/3) = 1/2$, the following must be true for particle $\boldsymbol{p}_k$ not to be visible.

$$\frac{\boldsymbol{v}_a \cdot \boldsymbol{v}_b}{|\boldsymbol{v}_a||\boldsymbol{v}_b|} \leq \frac{1}{2} \tag{4.37}$$

Without loss of generality, the vector $\boldsymbol{v}_a$ can be represented as a unit vector in the positive x direction (because of rotational invariance). Therefore $\boldsymbol{v}_a = [1\ 0]^T$ and the vector $\boldsymbol{v}_b$ is given by $[x - 1\ y]^T$ where (x, y) is the location of particle $\boldsymbol{p}_k$.

Equation 4.37 can be simplified as follows

$$\frac{x-1}{\sqrt{(x-1)^2 + y^2}} < \frac{1}{2} \tag{4.38}$$

The requirements, thus far, are that the particle $\boldsymbol{p}_k$ be located outside the circle of radius $\sqrt{3}$ and inside the region whose angle is $\pi/6$. This is stated mathematically in Equations 4.39 and 4.40.

$$\sqrt{x^2 + y^2} \geq \sqrt{3} \tag{4.39}$$

$$\frac{y}{x} \leq \tan\frac{\pi}{6} \tag{4.40}$$

The proof proceeds by using a contradiction method. Assume that the particle $\boldsymbol{p}_k$ is visible. It is then shown that one of the requirements in Equations 4.39 and 4.40 is violated. So the following is assumed:

$$\frac{x-1}{\sqrt{(x-1)^2 + y^2}} < \frac{1}{2} \tag{4.41}$$

which is to say particle $\boldsymbol{p}_k$ is visible to particle $\boldsymbol{p}_i$. Simplification of Equation 4.41 yields

$$\sqrt{3}(x-1) < y \tag{4.42}$$

but, from Equation 4.40, it can be shown that $y \leq \frac{\sqrt{3}}{3}x$. This, combined with Equation 4.42 results in

$$x < \frac{3}{2}. \tag{4.43}$$

Substituting this result with Equation 4.40 results in

$$y \leq \frac{\sqrt{3}}{2}. \tag{4.44}$$

Combining Equations 4.43 and 4.44 results in the following

$$x^2 + y^2 < \left(\frac{3}{2}\right)^2 + \left(\frac{\sqrt{3}}{2}\right)^2 = 3 \tag{4.45}$$

but this contradicts the requirement of Equation 4.39. Therefore, particle $\boldsymbol{p}_k$ is not visible by particle $\boldsymbol{p}_i$ because it is blocked by particle $\boldsymbol{p}_j$. Since $\boldsymbol{p}_k$ is blocked, its contribution to the update vector for $\boldsymbol{p}_i$ is zero. Therefore, $\boldsymbol{p}_i$ has a zero update vector regardless of other neighbors and the triangular lattice formation is steady. ∎

The reverse is not necessarily true. For example, a set of particles lying on a straight line with unit separation distance is steady by Definition 4.3.1. This is a result of the visibility blocking method.

Theorem 4.3.3 *The maximum size lattice neighborhood is 6.*

Proof The basic structure of the swarm is the equilateral triangle (not withstanding the example of a straight line given above) where the update vectors are all zero. The angles in an equilateral triangle are all $\pi/3$. Therefore, the maximum number of these structures that can be placed around a particle is $\frac{2\pi}{\pi/3} = 6$. ∎

4.3.2 Swarm States. Figure 4.25 shows a notional state transition diagram that might be used for a swarm application. These states are intended to capture the *global* behavior of the swarm, i.e., the entire swarm formation is in the state of *SEARCH*. With the exception of the states *INIT* and *FINISH*, all states imply some form of movement on the part of the swarm members. The *INIT* and *FINISH* states are shown for completeness. The *INIT* state might include the activities of deployment or taking off, and the *FINISH* state might include landing or completion. Failure conditions are not accounted for in this system, but should be in future analyses. State transitions that are bi-directional are indicated by heavier links.

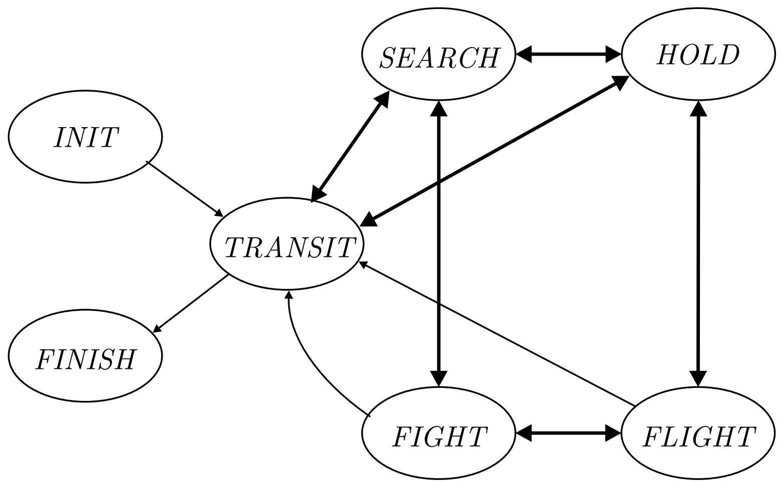


Figure 4.25 Global Swarm States

These states capture the *mission* of the swarm. However, this is a macro-view. It is hypothesized that each of these states can be accomplished with one of only two micro-level behaviors. These behaviors are defined as *coherent* and *incoherent*:

Definition 4.3.2 (Coherent) *The coherent behavior (CB) is obtained when the swarm moves in a* **coherent** *fashion, i.e., when the particle directions are all the same or very nearly the same (within* $\pm 45^o$*).*

Definition 4.3.3 (Incoherent) *The incoherent behavior (IB) is obtained when the swarm moves in a* **non-coherent** *fashion, i.e., when the particle directions are all different.*

Given these definitions, the states shown in Figure 4.25 can be categorized according to the manifested swarm behavior. This categorization is presented in Table 4.7. Since

Table 4.7 State Behavior Categorization

Coherent	***Incoherent***
TRANSIT	*SEARCH*
FLIGHT	*FIGHT*
	HOLD

the swarm behavior required for these states can be captured with two categories, only behaviors that match Definitions 4.3.2 and 4.3.3 are considered. Steadiness as defined applies almost directly to the *CB* behavior since the definition is made with respect to **no** relative particle movement.

4.3.3 Movement closure. Particle movements must fit within the vehicular capabilities of the sensor nodes. This section provides an analysis, independent of specific vehicle details, showing that movement is *closed*. Closure is formally defined in Definition 4.3.4.

Definition 4.3.4 (Closure) *Particle movement is **closed** if and only if velocity is bounded by some maximum.*

Definition 4.3.4 is not limited by particle *environment*. Here, *environment* is limited to the neighborhood of a particle. To show closure under this definition it is sufficient to show that the magnitude of particle position change, from Equation 4.28 is bounded by some finite value, as shown in Equation 4.46 where A_{max} is some constant.

$$|\Delta \boldsymbol{p}_i| = |\boldsymbol{p}'_i - \boldsymbol{p}_i| = \left| \alpha_s \left(\boldsymbol{v}_{dir} + \frac{\boldsymbol{v}'_{update}}{N_i} \right) \right| \leq A_{max} \tag{4.46}$$

The term $|\boldsymbol{v}_{dir}|$ is equal to unity for all particles. The term with $\boldsymbol{v}'_{update}$ is the composite update vector as described in Section 4.1.3.6. The adjustments to the direction of $\boldsymbol{v}_{update}$ of Equation 4.6 do not change the magnitude of the update vector. Therefore, the magnitude of $\Delta \boldsymbol{p}_i$ can be bounded as shown in Equation 4.47

$$|\Delta \boldsymbol{p}_i| \leq \alpha_s \left(1 + \frac{|\boldsymbol{v}_{update}|}{N_i} \right) \tag{4.47}$$

The analysis continues using stochastic methods. Consider the location of particles within each neighborhood region (see Table 4.4 and Figure 4.4) as a random variable. Then, the composite update vector $\boldsymbol{v}_{update}$ can be expressed as the sum of the mean vector components from each region as shown in Equation 4.48.

$$\boldsymbol{v}_{update} = w_{periph}(\alpha_1 \overline{w_d \boldsymbol{v}_1} + \alpha_2 \overline{w_d \boldsymbol{v}_2} + \alpha_3 \overline{w_d \boldsymbol{v}_3} + \alpha_4 \overline{w_d \boldsymbol{v}_4}) \tag{4.48}$$

The terms α_j, $j = 1, 2, 3, 4$ denote the proportions of the neighborhood set S_i that are in each region R_j. The symbol $\overline{\boldsymbol{v}_j}$ denotes the average of the update component from region R_j. This analysis assumes that w_{periph} is unity (i.e., peripheral vision is not a factor).

This assumption is appropriate since this analysis is seeking an upper bound on position updates. Further, the distance weighting factor w_d is distributed through the sum since it is a function of distance. This weighting is incorporated into finding the mean vectors for each *region* component. Equations 4.47 and 4.48 are combined to obtain Equation 4.49.

$$|\Delta \boldsymbol{p}_i| \leq \alpha_s \left(1 + \alpha_1|\overline{w_d \boldsymbol{v}_1}| + \alpha_2|\overline{w_d \boldsymbol{v}_2}| + \alpha_3|\overline{w_d \boldsymbol{v}_3}| + \alpha_4|\overline{w_d \boldsymbol{v}_4}|\right) \tag{4.49}$$

The quantities $\overline{w_d \boldsymbol{v}_j}$ are defined as the mean or expected value of the contribution to $\boldsymbol{v}_{update}$ for each region. The definition is given in Equation 4.50.

$$|\overline{w_d \boldsymbol{v}_j}| = E[|w_d(w_{attract}\boldsymbol{v}_{attract} + C_{align}\boldsymbol{v}_{align})|] \tag{4.50}$$

Equation 4.50 can be simplified and bounded—using the triangle inequality—as shown in Equation 4.51.

$$|\overline{w_d \boldsymbol{v}_j}| \leq E[w_d(w_{attract}|\boldsymbol{v}_{attract}| + C_{align})] \tag{4.51}$$

The magnitudes of $\boldsymbol{v}_{attract}$ and $\boldsymbol{v}_{align}$ are d and unity respectively (see Equations 4.10 and 4.11) for all four regions. Therefore, Equation 4.51 can be rewritten as shown in Equation 4.52. The weight $w_{attract}$ is a constant that depends on the region in which a neighboring particle is located.

$$|\overline{w_d \boldsymbol{v}_j}| \leq w_{attract}E[w_d d] + C_{align} \tag{4.52}$$

The expression $w_d d$ is treated as a random variable that represents the distribution of the contribution by other particles within a region. Without the w_d bias, it is assumed that particles are located within each region according to a uniform distribution. However, when particle locations are biased by w_d the mean location (specifically, the mean separation distance) is expected to be skewed *away* from the value obtained for the uniform distribution case. This is better explained by an example.

Consider the location of a neighbor particle within unit distance of $\boldsymbol{p}_i$ (i.e., $\boldsymbol{p}_j \in R_1$). The probability density function (PDF) with respect to separation distance is $p_1(d) = 2d$ with $d \in [0, 1]$. This result is obtained for uniform distribution over the *unit circle* and, since w_{periph} is assumed to be unity, is independent of the orientation of neighbor particles. The expected value in this case is simply $\frac{2}{3}$. The bias of w_d is incorporated by a form of its functional inverse as shown in Equation 4.53 and illustrated in Figure 4.26.

$$w_d = \sqrt{1-d} \rightarrow p_1'(d) \propto d^2 \tag{4.53}$$

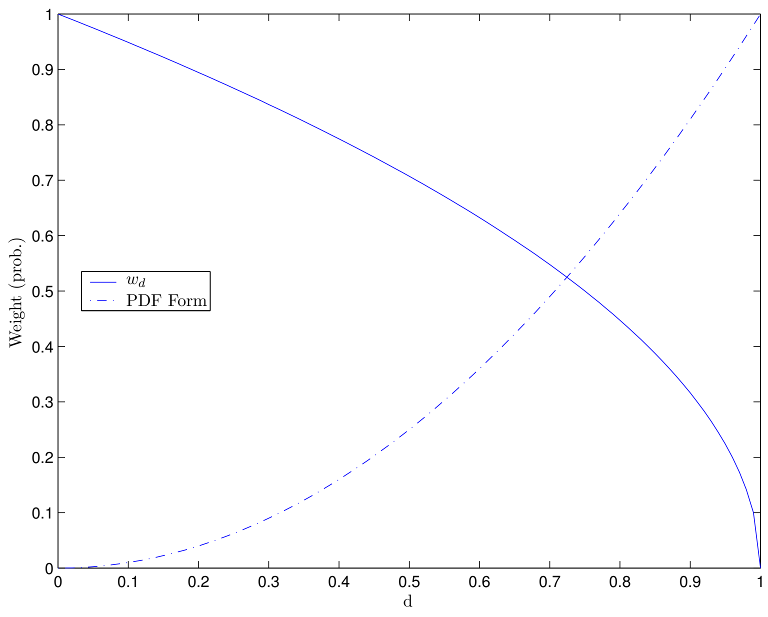


Figure 4.26 Region 1 Distance Weight and PDF

$$p_1'(d) \propto d(1-w_d) = d(1-\sqrt{1-d}) \tag{4.54}$$

Using this form and normalizing so that $p'(d)$ is a true PDF, Equation 4.55 is obtained.

$$p_1'(d) = 4d^3,\ 0 \leq d < 1 \tag{4.55}$$

The rationale in choosing $p_1'(d)$ in this fashion is based on the fact that w_d provides *emphasis* away from the center (location of $\boldsymbol{p}_i$) toward unit separation distance. This PDF results in a value of $\frac{4}{5}$ for $E[w_d d]$ which matches the intuition described above since $\frac{4}{5} > \frac{2}{3}$.

In a similar fashion, the PDFs and expected values for region R_3 can be found. These results are summarized in Table 4.8. Regions R_2 and R_4 are not included. The

Table 4.8 Distance Probability Distribution Functions

Region	PDF	Mean
1	$4d^3$	$\frac{4}{5}$
3	$d\sqrt{\frac{d_3-d}{d_3-d_2}}$	$\frac{191}{77}d_2$

weight $w_{attract}$ in Equation 4.52 is zero for region R_2. For R_4 it was determined that there are no particles ever in this region.

Using these results, a stochastic bound on $|\overline{w_d \boldsymbol{v}_j}|$ of Equation 4.52 can be obtained for each region. These results are summarized in Equations 4.56 through 4.58.

$$|\overline{w_d \boldsymbol{v}_1}| = C_{repulse}\left(\frac{4}{5}\right) + C_{align} \tag{4.56}$$

$$|\overline{w_d \boldsymbol{v}_2}| = C_{align} \tag{4.57}$$

$$|\overline{w_d \boldsymbol{v}_3}| = C_{repulse}\left(\frac{191}{77}d_2\right) + C_{align} \tag{4.58}$$

These equations are obtained from combining the results summarized in Table 4.8 and Equation 4.23. The sign on $C_{repulse}$ is not negative since the *magnitude* of the contribution is being used. Region R_4 is not included since there are no particles in this region.

With the stochastic characterization of the regional contributions to the update vector completed, the analysis of $|\Delta \boldsymbol{p}_i|$ can proceed. The results of the stochastic analysis are inserted into Equation 4.49 to obtain Equation 4.59.

$$|\Delta \boldsymbol{p}_i| \lesssim \alpha_s \left[1 + \frac{4}{5}\alpha_1 C_{repulse} + \frac{191}{77}\alpha_3 d_2 C_{attract} + C_{align}\right] \tag{4.59}$$

The term for region R_4 is left off since its contribution is statistically zero. The symbol $\lesssim$ is used in place of $\leq$ since the bound no longer strictly holds but holds in a stochastic sense.

As Equation 4.59 shows the update distance of each particle can be bound (in a stochastic sense). This implies that, with proper choice of parameters, particle movement is closed. This result is used in Section 7.1.4 to show what the bounds are for the coherent and incoherent behaviors.

4.3.4 Neighborhood size.

It is conjectured that position updates, which are dependent on neighborhood sizes, are independent of the overall swarm size. This conjecture is based on the vision model described in Section 4.1. However, preliminary results show that this is not the case–this section explains why this is so and provides an analytical explanation.

The error in reasoning for the above conjecture is that it assumes that the neighborhood sizes are constant for all particles regardless of swarm size. This is not the case. The following analysis develops a method for determining the average neighborhood size. This is compared to experimental results of actual neighborhood sizes obtained for several swarm sizes. In order to do this, a model for the physical configuration of a swarm formation is developed. This model is used to obtain a formula for predicting the average particle neighborhood size.

It is assumed that the shape of the swarm at any given time–with an ideal separation distance of unity–is circular with radius R. This illustrated in Figure 4.27 where $R = 3$. The number of particles as a function of the radius of the circle is approximated

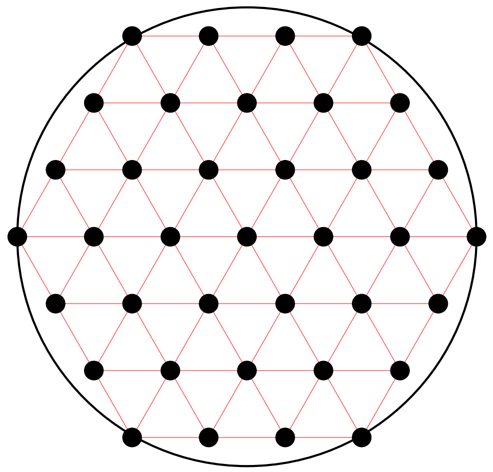


Figure 4.27 Circular Approximation

by Equation 4.60 where N is the size of the swarm and $L = \lfloor 2R/\sqrt{3} \rfloor$.

$$N(R) \approx 2R + 2\sum_{j=1}^{L} \left\lfloor 2\sqrt{R^2 - \left(j\frac{\sqrt{3}}{2}\right)^2} \right\rfloor \tag{4.60}$$

A plot of N for several values of R is shown in Figure 4.28. The challenge is to find an expression of R in terms of N. However, solving Equation 4.60 for R is not possible. Therefore, an approximation is determined to facilitate finding an analytical expression for R. The curve in Figure 4.28 can be approximated with Equation 4.61.

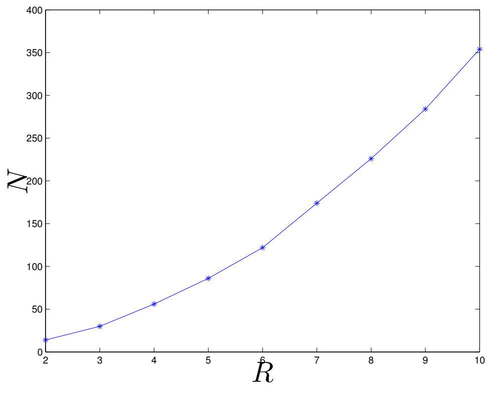


Figure 4.28 Swarm Size

$$N \approx AR^2 \tag{4.61}$$

The constant A was found to be 3.506 by computing the weighted average of the quotient N/R^2 using the values of N from Equation 4.60. By solving Equation 4.61 for R an approximation of the formation radius may be obtained. The shortcoming of the above analysis assumes that the number of particles as a function of the radius R is continuous. This is clearly not the case. The data points of Figure 4.28 are connected by a continuous line. In reality, the function $N(R)$ is highly discontinuous because of the floor function. However, it is felt that the error is small enough to make the approximation in Equation 4.61 sufficiently useful for the subsequent analysis. Therefore, the radius of the formation as a function of the swarm size N is as shown in Equation 4.62 where $B \approx 0.5341$.

$$R \approx \sqrt{\frac{N}{A}} = B\sqrt{N} \tag{4.62}$$

The average neighborhood sizes for two scenarios—one coherent behavior denoted by the solid line and one incoherent behavior denoted by the dashed line—are plotted in Figure 4.29. Not surprisingly, the average neighborhood sizes for the two scenarios are similar to each other but clearly vary as a function of the total swarm size. However, the

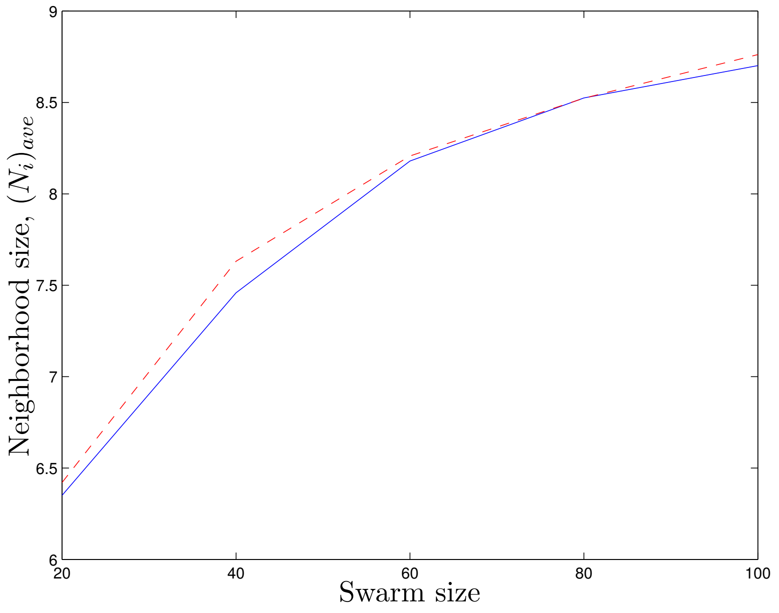


Figure 4.29 Circular Approximation

standard deviation for the two scenarios differ significantly but is close to constant for all swarm sizes. The standard deviation for the coherent behavior and incoherent behavior scenarios is 1.0916 and 1.9003 respectively.

The neighborhood size varies as a result of the particles on the edge of the formation. It is assumed for this analysis that each particle on the edge has a neighborhood size that is approximately one half of that for other particles[5]. This concept is presented in Equation 4.63.

$$(N_i)_{ave} = (1-\beta)N_i + \beta\frac{N_i}{2} \tag{4.63}$$

The term β in Equation 4.63 is the proportion of particles on the edge of the formation. An approximation for β is simply the circumference of the swarm formation. Since the formation is assumed to circular, the number of particles on the edge can be obtained from Equation 4.64.

$$N_{edge} \approx \lfloor 2\pi R \rfloor \tag{4.64}$$

[5]In practice it is expected that the neighborhood size would be slightly less than one half. This is due to the convex nature of the swarm formation edge.

For the sake of analysis, the floor operation is dropped. Then β can be expressed as shown in Equation 4.65.

$$\beta \approx \frac{N}{2\pi R} \tag{4.65}$$

Combining Equations 4.62 and 4.65 with Equation 4.63 an expression for $(N_i)_{ave}$ can be obtained as shown in Equation 4.66.

$$(N_i)_{ave} = \left(1 - \frac{\sqrt{N}}{2\pi B}\right) N_i + \left(\frac{\sqrt{N}}{2\pi B}\right) \frac{N_i}{2} \tag{4.66}$$

Normalizing Equation 4.66 with respect to the *constant* neighborhood size N_i and simplifying obtains Equation 4.67.

$$\frac{(N_i)_{ave}}{N_i} = 1 - \frac{\pi B}{\sqrt{N}} \tag{4.67}$$

Plots of the experimental and theoretical results are shown in Figure 4.30. The curve

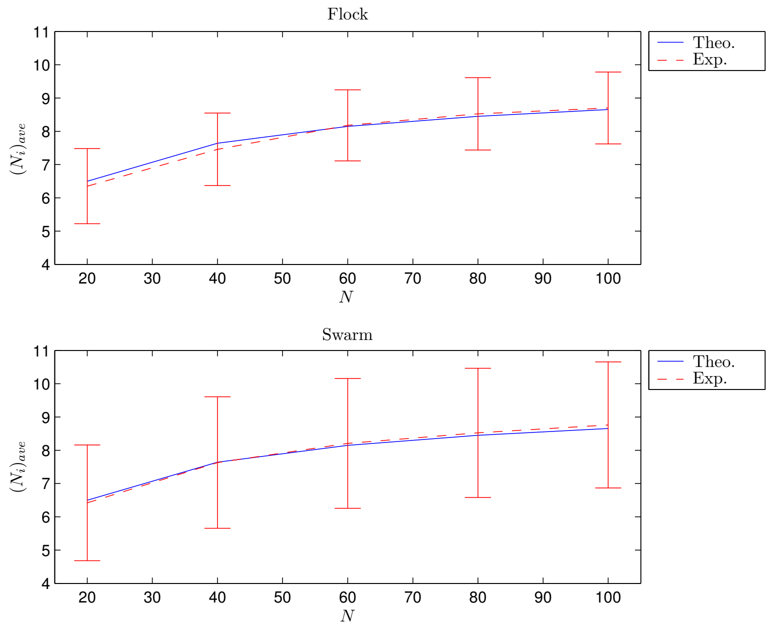


Figure 4.30 Theoretical and Experimental Neighborhood Size

for the experimental data includes error bars for one standard deviation at each data point. The curve for the theoretical data is obtained for $N_i = 10.3867$. This value is obtained by finding the average of the ratio of the theoretical result of Equation 4.67 and the experimental data for the flock and swarm scenarios. The standard deviation for N_i

is 0.1393. Given this and the standard deviation of the experimental data, the theoretical and experimental results compare well.

There are several conclusions to be drawn from these results. First, the effect of the edge particles on the average neighborhood size is a plausible cause for the variation. Second, as the first conclusion implies, the neighborhood size is constant regardless of overall swarm size and behavior. Third, the value of the neighborhood size is not 6 as is shown above in Theorem 4.3.3–the estimated value is approximately 10.4. A probable explanation for this is due to the small perturbation that is introduced by the movement of the particles (see Equation 4.26). Theorem 4.3.3 assumes zero perturbation and a zero comfort zone factor.

4.4 Summary

This chapter presents a detailed description of a formal swarm model. The model incorporates innovative peripheral vision and vision blocking mechanisms that improve the fidelity of the model. The use of vision blocking makes the model scalable so that, for a chosen behavior (and the associated parameter set values), changing the number of particles has no affect on behavior.

Additionally, an in-depth discussion of swarm parameters affecting such things as peripheral vision weighting, alignment, attraction, repulsion, comfort zone, speed adjustments, and turn perturbations is presented. Three important properties of the swarm model are demonstrated. First, it is shown that the swarm formation is steady. This is an important result for predictability with respect to formation characterization. Second, it is shown that the model provides for movement closure. This is important since real world systems cannot move with infinite (or even relatively large) speeds. Third, it is shown theoretically and validated by experiment that position updates, which are dependent on neighborhood sizes, are independent of the overall swarm size.

Finally, a connection is made between swarm behavior–the globally observable formation characteristics–and potential swarm formation states. The states relate to swarm function, e.g., *search*, *flight*, or *hold*. It is proposed that the two swarm behaviors, *coher-*

ent and *incoherent*, provide the correct type of formation dynamics for each state. These formation dynamics–with the time-varying neighborhood sets–provide the challenge for swarm-based sensor networks as discussed in Section 1.1. The next chapter discusses several communications models that are designed to address the efficiency and effectiveness issues associated with the dynamics of these networks.

V. Networking and Swarm Based Sensors

The challenge for routing protocols in the dynamic environment of sensor swarm networks is to balance the need for efficiency with effectiveness. Protocol efficiency deals with bandwidth utilization while effectiveness deals with successful delivery of information throughout the system. Three routing protocols are presented here that range in their efficiency and effectiveness. Other protocols exist–as described in Section 3.2–and include AODV, DSDV, SLURP, and GLS. However, these protocols are not well suited for the dynamics of swarm-based sensor networks.

One of the protocols described in this chapter establishes the baseline for the efficiency and effectiveness measures. A flooding protocol is usually highly effective because of the high redundancy but very inefficient. The other two protocols–Geographical-based routing and Directed Diffusion–incorporate heuristics to significantly improve efficiency while not reducing effectiveness. The performances of these protocols in the dynamic sensor swarm network environment is described in Section 7.2.

This chapter details the routing protocols that were developed and investigated for the swarm-based sensor system. Analysis of theoretical and simulation performance is also provided. As stated earlier in Section 2.4.2, the *ns-2* simulator is used to simulate the network communications environment. A detailed description of the *ns-2* simulator and a performance comparison between *ns-2* and OPNET are included in Appendix C. Before describing the details of the three protocols, a brief description of the communications model for the swarm-based sensor network system is given.

5.1 Swarm-based Sensor Network System

The general mobile node model, shown in Figure 5.1, consists of elements equivalent to those in the Open Systems Interconnect (OSI) network layer model developed by International Standards Organization (119) (see Section 2.4.1). The Generic Network Interface of Figure 5.1 includes the physical and Medium Access Control (MAC) layers of the OSI model and is described in detail in Section 5.1.1 (as well as Figure 5.2). Potential MAC layers include the IEEE 802.11 system (85), the IrDA system (57), and the Bluetooth

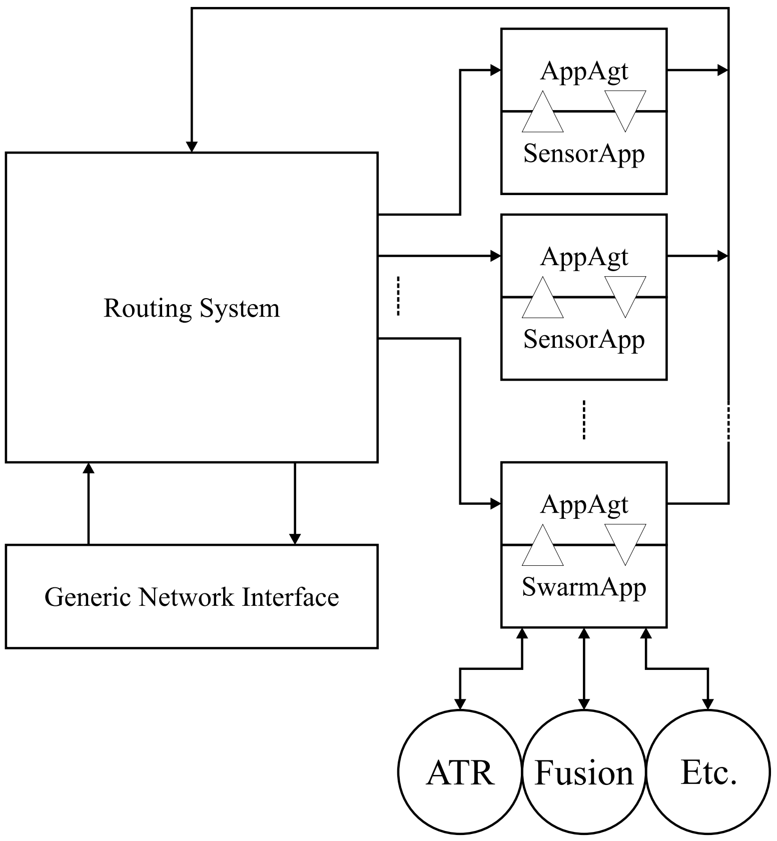


Figure 5.1 Mobile Node Model

system (13). For this implementation, an 802.11 MAC model is used in broadcast mode to provide the channel interface. The "Routing System" block of Figure 5.1 contains the details of the three routing protocols. Each protocol is developed with standard interfaces to the Generic Network Interface as well as the sensor swarm application interfaces. The interface elements are described in detail in Section 5.1.2. Basically, these interfaces provide a standard access to the communications network for sensor applications and *data analysis* applications.

5.1.1 Communications Model. The channel interface components consist of standard model elements from the respective modeling systems. The radio propagation model is based on a simple, line-of-sight range model. The antenna model is omnidirectional with unity gain. The MAC model is the 802.11 wireless LAN model used in broadcast mode with a data rate of 11Mbps. These elements are shown in Figure 5.2. The link layer com-

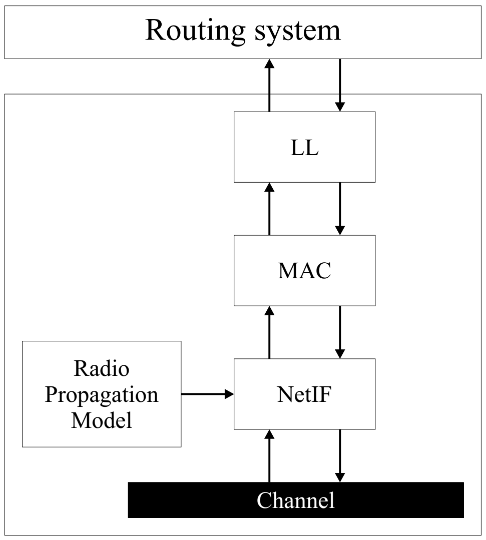


Figure 5.2 Network Interface Components

ponent (*LL*) provides the interface between the routing system (as shown in Figure 5.5) and the lower level elements. It also provides queuing support for packets that get deferred by the MAC. The network interface component (*NetIF*), along with the radio propagation model, provide the physical access to the shared channel.

It should be noted that the broadcast mode of 802.11 does not guarantee successful communications. This is because no link negotiations–such as *Ready-to-send* and *Clear-to-send*–are used (56). This mode causes competing issues. Broadcast mode allows a single packet to be received by multiple nodes simultaneously and hence, a gain in efficiency. However, because there are no negotiations, delivery is not guaranteed and delivery status is unknown which represents a potential loss of effectiveness.

5.1.2 Swarm Applications. The swarm-based sensor network system provides support for a distributed sensor processing system wherein nodes possess sufficient sensory and processing capability to perform some high level computational tasks. These tasks might include (as shown in Figure 5.1) automatic target recognition (ATR), event notification, or sensor fusion. These comprise the swarm applications that reside within the network. The (*SwarmApp*) component provides a generic interface to the sensor swarm network system for all applications. The sensors are represented by the sensor application

(*SensorApp*) components. These sensors are abstract representations (in this model) of high-fidelity, large data producing elements such as image, infra-red, or synthetic aperture radar (SAR) sensors. Each node may have one or more sensors.

Each application interfaces with the node via an application agent (*AppAgt*). The *AppAgt* component provides the interface between the packet-based routing system and the abstract data generated and processed by the applications. Once data is encapsulated in packets by the *AppAgt*, the packets are forwarded to the routing system for subsequent forwarding to other nodes or local applications.

5.1.3 Application Messages. Two types of messages are generated by the applications in the sensor network system. Swarm applications generate *requests* for sensor data, and sensor applications generate *responses*. Requests can include several parameters that specify such things as the type of sensor, the location of interest (point or region), and the time (instant, duration with start/stop times, or infinite). A request message is forwarded from the originating node to all other nodes. Response messages are forwarded via the routing protocol from all nodes to the node (or nodes) that requested the sensor data. This is demonstrated in Figure 5.3. The response message includes the sensor data

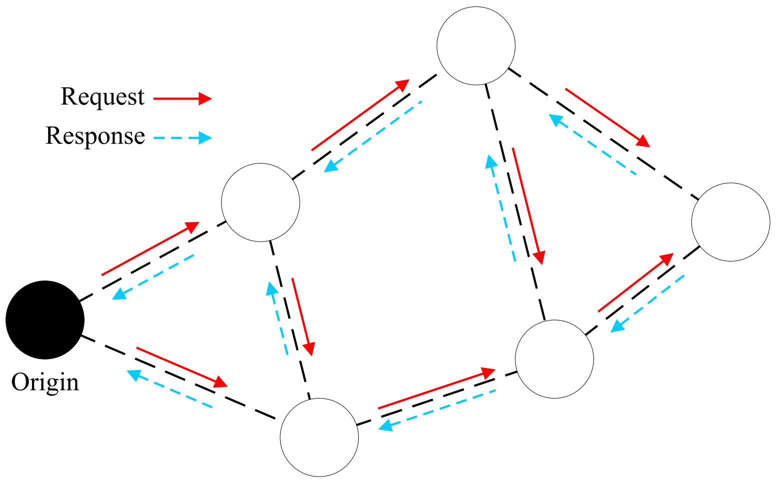


Figure 5.3 Message Propagation Example

as well as a record of the request message that it satisfies. For the purposes of this research effort, request messages specify a global scope and have no expiration. This ensures that, once a node receives a request message, all its sensor data (in the form of response messages) are propagated through the network to the request origination node. It should be noted that, for the purposes of this work, request messages are originated by specific nodes. For a realistic implementation of the swarm-based network system, requests for sensor data would originate from a source external to the swarm of sensors and be input into the swarm from any node (see Section 2.2).

Three routing protocols are described here. A flooding protocol is developed to establish a baseline for routing performance in the context of a swarm sensor network. A simple, stateless location based protocol is developed–the conjecture is that state-based protocols would be unable to react quickly enough to the dynamics of a swarm system. Therefore, a simplified Directed Diffusion (58) protocol is developed. Each of these three protocols are described in greater detail beginning with Section 5.2.

5.1.4 Node Movement. Node movement in *ns-2* is supported directly via specialized procedures of the simulation kernel and mobile node objects. The swarm-track binary data need only be converted to an OTcl script and read in at simulation time. A simple utility, written in C and described in Appendix I, is used to generate the text output of the OTcl movement script.

In *ns-2* nodes are instantiated in a simple loop with node locations and movement specified as part of the simulation script. Figure 5.4 shows a portion of the simulation script that accomplishes this task in order to illustrate the methodology for accessing the *ns-2* kernel. The variable `$ns_` provides access to the simulation kernel. The `node` kernel

```
for {set i 0} {$i < 4} {incr i} {
  set node_($i) [$ns_ node $i]
  set src($i) [new Application/SensorApp]
  $node_($i) add-sensor $src($i)
}
```

Figure 5.4 GRP Node Instantiation

procedure is used to instantiate an array of nodes (in this case 4) as well as instantiate a sensor application for each new node. Node position is set explicitly by assigning values to a node's location variables (`X_`, `Y_`, `Z_`) as follows:

```
$node($i)_ set X_ 4.0
$node($i)_ set Y_ 8.0
$node($i)_ set Z_ 0.0
```

Node movement is accomplished by specifying a destination point (in 2 dimensions) and a speed as follows:

```
$node($i)_ setdest 25.0 13.0 2.0
```

This causes the node to 'move' toward the location (25, 13) at a speed of 2 distance units per second. Clearly, node movement is conducted in linear segments. If the segments are short, smooth motion can be approximated. Nevertheless, linear interpolation is used to obtain node positions at times when nodes are between segment points.

5.2 Geographical Routing Protocol

The Geographical Routing Protocol (GRP) is a naive, stateless, location-based protocol much like that presented in (40) but with node mobility. It is similar in its intended use to other ad hoc routing protocols such as *Directed Diffusion* (28, 58) and the *Geographical Addressing and Routing Protocol* (28, 58) as described in Sections 3.2.2 and 3.2.3.

As stated earlier, the swarm model is defined to operate in two-dimensional space. For ground based mobile sensors this is sufficient, but for airborne sensors this might be inadequate. However, it is assumed that airborne sensors are tasked with observing regions/events/activites on the ground. Because of this, a vertically oriented formation of sensors provides only a redundant view of the same location on the ground. The second assumption is that the receivers/transmitters used provide line-of-sight access and reception is based on the free-space model (39). In other words, if two nodes (one transmitting, another receiving) are within a specified range, then packets are received correctly. However, this model is sufficiently complex to handle collisions associated with the hidden and exposed station problems (119).

The components of the GRP system are shown in Figure 5.5. Descriptions of these components are provided in the following sections. There are two types of messages in

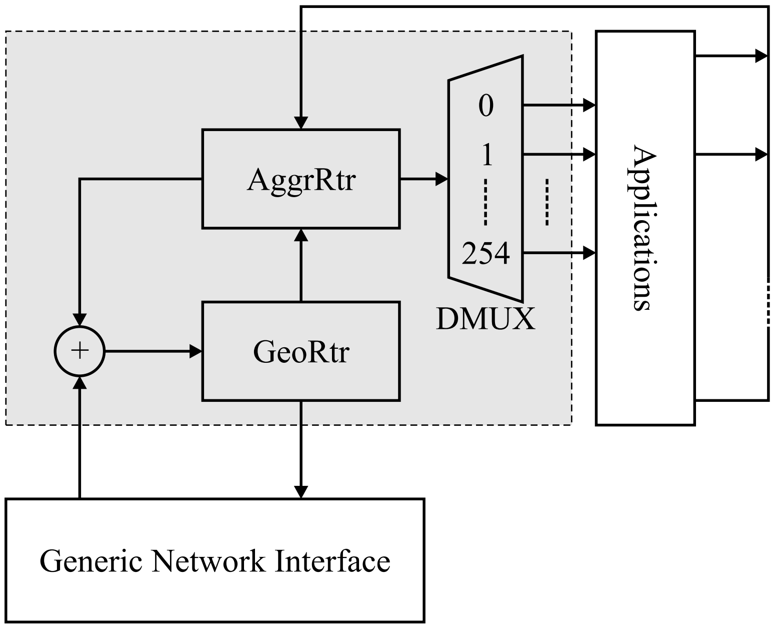


Figure 5.5 GRP Node Model

the GRP system. The *RQST* message is used to propagate a request for some type of information, and the *RESP* message is used to encapsulate the data that provides a response to one or more RQST messages. Both types of messages are encapsulated in packets which provides the basic communications element of the system. One message is always encapsulated by a single packet.

5.2.1 Message Handling.

Packets encapsulating RQST messages are handled differently than packets for RESP messages. Since the desire is to propagate RQSTs to all nodes, vector-based routing calculations are performed according to Equation 5.1 and as illustrated in Figure 5.6.

$$D_{RQST} = (\boldsymbol{p}_r - \boldsymbol{p}_{orig}) \cdot (\boldsymbol{p}_r - \boldsymbol{p}_t) \tag{5.1}$$

In Figure 5.6, the node originating the RQST is denoted by $\boldsymbol{p}_{orig}$ while the receiving and transmitting nodes are denoted by $\boldsymbol{p}_r$ and $\boldsymbol{p}_t$ respectively. If the result, D_{RQST}, of the vector dot-product in Equation 5.1 is positive (Fig. 5.6a) then the receiving node *accepts*

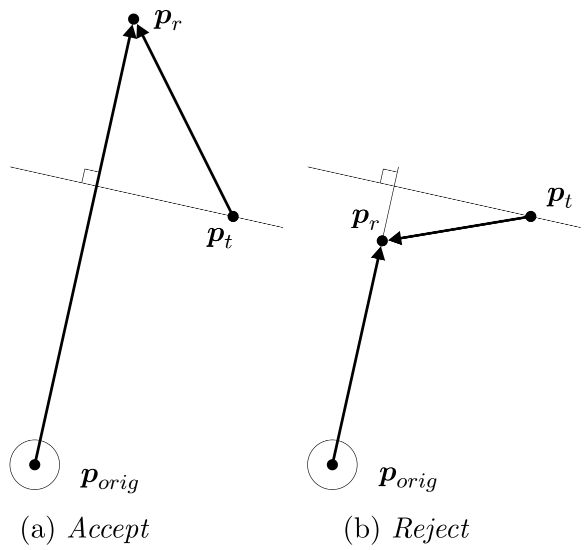


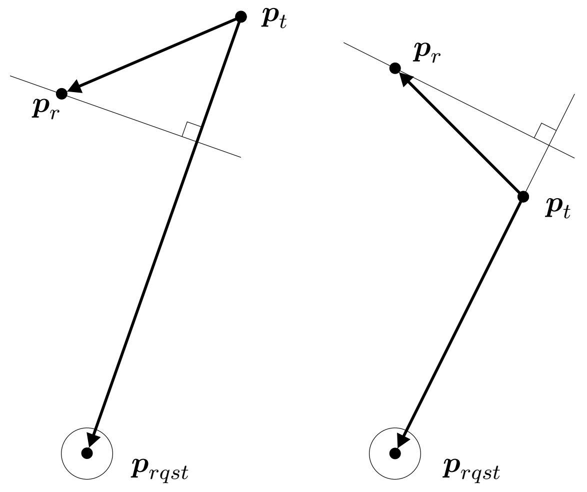
Figure 5.6 RQST Packet Routing

the incoming packet for further processing (aggregation and/or retransmission). If the result is negative (Fig. 5.6b), the packet is dropped. Additionally, the *GeoRtr* keeps a record of *accepted* packets. This eliminates the potential for circular routing and reduces redundancy due to multi-path routing of the same packet. This routing method for RQSTs provides a ripple effect much like the ripples generated in a body of water by a dropped object.

Packets encapsulating RESP messages are handled as follows. First, a RESP message is only generated if there exists a RQST for the data. More details on this are given in Section 5.2.3 where the *AggrRtr* component is discussed. Assuming a record of one or more RQST messages exist at the active sensor node, a RESP message is generated and forward in accordance with the rules for RESP messages. Information for each pending RQST is embedded in the outgoing packet. This information includes location of the RQST originating node (at the time the RQST was generated) as well as other unique identifier data (the node and RQST packet identifiers).

There are two possible methods for handling multiple pending RQSTs–send a separate RESP for each RQST or send one RESP with info for multiple RQSTs embedded in the RESP packet. The former method provides a simpler implementation at the expense of increased network loading. The latter reduces redundancy but increases the complexity of processing RESP packets. This is because a dynamic structure to store multiple RQST elements must be embedded in the RESP packets and the router must deal with the potential for multiple destinations. While the latter method is more complex, it was determined that the channel resource was sufficiently limited as to benefit from a reduced network load.

The mathematics for the *GeoRtr* with respect to RESP packets is similar to RQST packets. This is shown in Figure 5.7 and Equation 5.2 where $\boldsymbol{p}_{rqst}$ is the location of the node that originated the RQST message.



(a) *Accept* (b) *Reject*

Figure 5.7 RESP Packet Routing

$$D_{RESP} = (\boldsymbol{p}_{rqst} - \boldsymbol{p}_t) \cdot (\boldsymbol{p}_r - \boldsymbol{p}_t) \tag{5.2}$$

The difference lies in the direction of the vectors. The condition of acceptance is the same: if the result of the dot product D_{RESP} is positive (Fig. 5.7a) then the RESP packet is processed for potential aggregation and retransmission. Otherwise it is dropped (Fig. 5.7b). Specific details concerning packet format and management are given in Section 5.2.5.

5.2.2 Geographical Routing Element. The *GeoRtr* component provides the lowest level routing for the sensor network system. Since packets are transmitted by the MAC in broadcast mode, all nodes within range of a transmitting node receive the outgoing packets (provided there are no collisions). The *GeoRtr* ensures that packets are retransmitted only when it is correct to do so. Correct handling for RQST and RESP messages is defined in Section 5.2.1.

5.2.3 Aggregation Router. While the *GeoRtr* provides routing support at the physical location level, the *AggrRtr* provides support at the RQST/RESP message level. The *AggrRtr* maintains two lists of pending RQST messages–those generated locally and those remotely generated and received via the network. It also provides the mechanism by which packets that contain redundant information can be aggregated into a single packet and thus reduce the network load. This component also implements the algorithm for forwarding RESP messages (both locally and remotely generated) to a resident application in response to a RQST message.

5.2.4 Demultiplexer. The remaining component of the routing system to be discussed is the demultiplexer (*DMUX*) as shown in Figure 5.5. The DMUX is used to route RQST and RESP messages to the appropriate application. All RESP messages get routed to the single *SwarmApp* application (if it exists) via its *AppAgt* interface. The application itself determines what function (or multiple functions) get the sensor data contained in the RESP message. The *AppAgt* for *SwarmApp* applications is always installed on a single *port* (in this case 254). On the other hand, multiple sensors of different types may be installed on different nodes (while the nodes are homogeneous with respect to routing capabilities and mobility, the sensor suite on each node is not required to be homogeneous). As sensor applications are installed on a node, they are assigned a unique port by the *DMUX*. This method of multi-application access is patterned after the *ns-2* method (39).

5.2.5 Packet Management. A specialized packet format is used to carry the RESP/RQST message data and the necessary header information to support the geographical vector calculations. Details of the packet format, along with field descriptions, is given in Table 5.1.

Table 5.1 GRP Packet Header Format

Name	Description
`TYPE`	Type of packet (RQST, RESP)
`SRC`	Originating node identifier
`UID`	Unique packet identifier
`LAST`	Node identifier for last transmitting node
`S_PORT`	Port number for originating application
`N_RQSTS`	number of 'matching' RQSTs
`RQSTS`	data structure containing RQST info
`x, y`	location of originating node
`tx, ty`	location of transmitting node

The `TYPE`, `SRC`, `UID`, `LAST`, `S_PORT`, (`x, y`), and (`tx, ty`) fields are initialized by the *AppAgt* for new packets (all packets are created by the *AppAgt*). The *AggrRtr* fills in the `N_RQSTS` and `RQSTS` fields for new RESP packets with all pending RQSTs. The *GeoRtr* updates the `LAST` and (`tx, ty`) fields.

The `RQSTS` field is a dynamic structure that contains the following information for each pending RQST: `SRC`, `UID`, and (`x, y`). The location information pertains to the node originating the RQST at the time of origination. This information is used by the *GeoRtr* to correctly route RESP packets toward the RQST originating node and by the *AggrRtr* to forward received RESP packets to the appropriate application via the *DMUX* and *AppAgt* components.

5.2.6 Data Traffic. As stated in Section 5.2, RQST messages are originated by specific nodes. For the test scenarios used here a single node generates one RQST message. Sensor RESP messages are generated with exponentially distributed inter-arrival times. The average inter-arrival time is 2 seconds. This value was chosen in order to ensure that there were not multiple RESP messages propagating through the network at any given time.

5.3 Directed Diffusion Routing Protocol

The Directed Diffusion protocol is described in detail in (58) and in Section 3.2.2. The implementation of Directed Diffusion for a swarm-based sensor system is described here. A simplified version of the protocol called sDIFF is used for this research effort.

Section 3.2.2 provides an overview of the protocol. This section describes the specific implementation details for the sensor swarm network system. The structure of the routing system is similar to that of the GRP system. Figure 5.8 details the components that make up the routing system. There are three message types in the sDIFF routing system. An

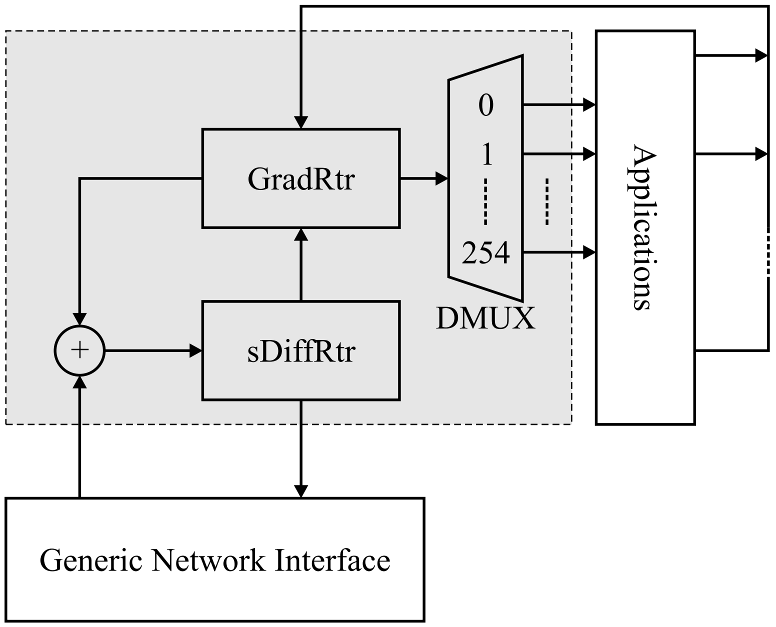


Figure 5.8 GRP Node Model

INTR message is used to express an *interest* in some type of sensor data. The response to the interest is contained in a *DATA* message. The *DRDY* message is the third type and is used for internal communication sensor status for routing elements. The *INTR* and *DATA* messages are equivalent to the GRP *RQST* and *RESP* messages respectively. The symbology of *INTR* and *DATA* is used here to be consistent with the description of the protocol in Section 3.2.2 and (58).

5.3.1 Simple Diffusion Routing Element.

The *sDiffRtr* is the lowest level component of the *sDIFF* system. It is less complex than the *GeoRtr* element in the GRP system since it only ensures that cycles do not occur when packets are routed in the network–it provides no direction checking mechanism. Cycles are avoided by maintaining a list identification records for successfully received packets. If a received packet already has an entry in the record list, then it is quietly discarded and no subsequent processing is performed.

5.3.2 Gradient Router. The gradient router element *GradRtr* manages the *gradients* used to control routing. This component is responsible for sending updates at the required rates based on the *reinforcements* that it receives. It also degrades routes based on a lack of reinforcements.

The *GradRtr* maintains two lists for *INTR* packets–one for locally generated requests and the second for remotely generated requests. The `DiffLCache` and `DiffRCache` objects encapsulate these two lists. The underlying data structure for both objects is a linked list. The forms of these list elements are shown below. There are a number of common elements between the two objects. These are listed and described in Table 5.2.

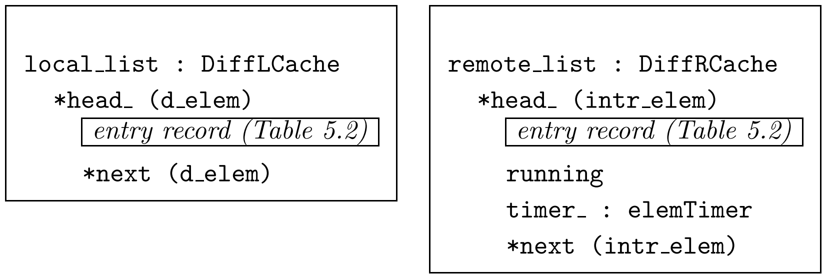


Figure 5.9 GRP Node Instantiation

The difference between the two lists is the `running` and `timer_` variables used by the remote list. These are explained below.

Table 5.2 Cache Record Entries

Name	Description
`id`	Unique integer identifier
`type`	Type of interest
`rate_orig`	Original requested data rate
`duration`	Duration in seconds of the *interest*
`timestamp`	Time stamp for last received *interest*
`expiresAt`	Time when the *interest* expires
`*data_list`	List of matching *DATA* packets
`numN`	Number of neighbors with this *interest*
`N[]`	Array of neighbor records

With the exception of the `*data_list`, `numN`, and `N[]` variables, the entries in Table 5.2 match the purpose and description given for Directed Diffusion in Section 3.2.2. Briefly, the `type` variable specifies the type of object (tank, elephant, etc.) that the particular interest is for. The `rate_orig` variable defines the desired update rate (in updates per second). The `duration`, `timestamp`, and `expiresAt` variables are used to determine how long the interest remains valid in the system. Once an interest expires, data updates are no longer provided. For the scenarios used in this research effort, the duration is set so that the interests do not expire.

The `data_list` variable is used to keep a list of received *DATA* packets that satisfy the interest. This data structure is encapsulated in the `DiffPktList` object. This object provides methods for accessing the list of packets.

The `numN` and `N[]` variables provide a mechanism for maintaining a history of neighboring nodes that expressed an interest that matches the particular entry. This information is used to selectively reinforce those neighbors that provide data updates. The neighbor information is an array of records containing information as shown in Table 5.3. A new entry is added to the array when an interest–with a previously existing record in the `local_list` or `remote_list` variables–is received. The `out_rate` variable is used by `remote_list` to maintain the outgoing reporting rates for each node's neighbors, and `in_rate` is used by `local_list` to maintain the input rate for locally generated interests. The entries are updated as described in Sections 5.3.2.1 and 5.3.2.2 for output and input rates respectively.

Table 5.3 Neighbor Record

Name	Description
`id`	Unique node identifier
`out_rate`	Output data rate
`in_rate`	Input data rate
`ts`	Last time of rate update
`last_data`	Last time of data match

5.3.2.1 Output Rate Updates. The discussion of input and output rates begins with the *sink* node. As stated in Section 3.2.2, the *sink* node is the first node in the network to receive external tasking. In addition to specifying the data needed (e.g., type, location), the original tasking comes with a desired data rate. The sink node sends out an initial interest message with a much lower data rate. In order to not overload the network, a maximum data rate was set to be 10 samples per second. The initial low data rate was set at $\frac{1}{10}$ of the maximum data rate but not less than 1 sample per second. Issues associated with what value is used to scale the initial data rate are discussed in Section 3.2.2.

Interest messages propagate through the network in one of two ways. Initially, messages are routed using a flooding mechanism. The only routing function performed is the elimination of cycles. The second method uses a form of "multicast" to send interest messages to a subset of the neighborhood node. While the physical layer still uses a broadcast method, the packet header includes routing information to indicate the intended nodes. This is described in greater detail in Section 5.3.4.

The interest message causes any data sources (sensors) to begin producing data at the initial low data rate. Additionally, the interest messages are cached by each node. The interest cache–`remote_list`, as described earlier–can have multiple entries, one for each *different* interest. In this work only one interest type is considered for purposes of simplicity.

As nodes receive the interest messages, entries are added to the node neighbor array `N[]`. An example record array for a node with neighbors A, B, and C is shown below.

Node	A	B	C
`out_rate`	1	1	1
`ts`	1.0	1.1	0.9
`last_data`	0.0	0.0	0.0

The `ts` variable indicates the notional wall-clock times when the last interest update was received from each respective node. The `last_data` values are initialized to all zeros.

The `out_rate` is used to determine whether data packets need to be forwarded or not. For instance, say the following record set exists after some time.

Node	A	B	C
`out_rate`	1	2	2
`ts`	3.8	4.1	4.2
`last_data`	3.4	3.6	3.7

The *need* time t_{need} for each neighbor is computed according to Equation 5.3 where t_{last_data} is `last_data`.

$$t_{need} = t_{\texttt{last\_data}} + \frac{1}{\texttt{out\_rate}} \tag{5.3}$$

This data set results in the *need* times for each neighbor node:

Node	A	B	C
t_{need}	4.4	4.1	4.2

If a data packet is received with a time stamp of $t = 4.3$ then the data packet would be forwarded to nodes B and C and the record set would be updated as:

Node	A	B	C
`out_rate`	1	2	2
`last_data`	3.4	4.3	4.3

For interest message updates, the existing `out_rate` value is incremented. This is done independently of the requested reporting rate encapsulated in the interest message. As stated earlier, a new entry is added when a new neighbor is "discovered" for a previously existing interest message. The new entry is initialized with an output rate that is a fraction of the require rate. Since interest messages propagate outward from the sink node in a flood fashion, this mechanism provides a method for informing all nodes of the interest. Then, as nodes produce data messages, the more efficient routes can be reinforced to improve throughput while maintaining a low level of redundant background traffic to allow for adaptation to network dynamics. The issues concerning the scaling fraction is discussed in Section 3.2.2. The value chosen is $\frac{1}{10}$. The minimum data rate is set such that it is never less than 1 sample per second. The initial maximum data rate requested is set to 10.

5.3.2.2 Input Rate Updates. The input rate is maintained by a sink node in the `local_list` cache. It is used to generate new interest messages in order to reinforce the more efficient data routes by causing increases to the data rates for selective nodes.

The `in_rate` value for a particular node entry is updated when a data message is received that matches an interest entry in the `local_list` cache. However, only the node from which the first data message is received is updated. This is illustrated in Figure 5.10 where node R receives a data message, originating by node F, from nodes E, F, and G. However, since the messages from E and G travelled a longer route, only the entry in the neighbor array `N[]` for node F is updated. The data messages received from E and G are discarded. An example is shown below. The updated `in_data` values are:

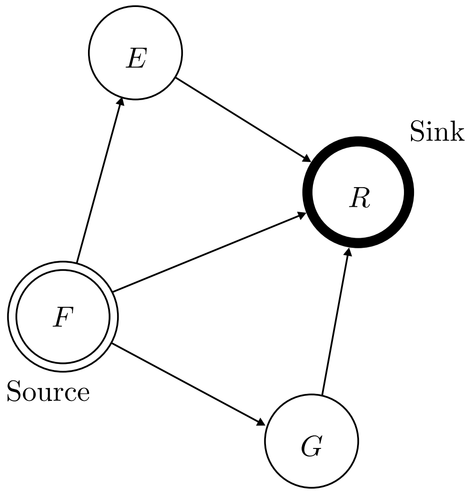


Figure 5.10 Data Message Routing

Node	E	F	G
`in_rate`	1	2	1
`ts`	8.1	7.8	8.3
`last_data`	6.9	7.7	7.0

The update causes the `in_rate` to be incremented by one for each successfully received data message. In addition to the increment, a reinforcement interest message is sent to the node from which the data message was received–in this case node F.

The reinforcement interest message is broadcast by node R but the packet header encodes node F as the only intended recipient. This causes nodes E and G to discard the packet. Node F updates its `remote_list` cache as described in the previous section. Node F then checks its data cache to see which of its neighbors should be reinforced. The neighbor from which the last data packet was received is selected for reinforcement. A new interest message is sent to this node using the "multicast" method.

5.3.2.3 Route Degradation. In order to handle network dynamics, routes that do not produce regular data updates are degraded. Each node maintains several timers that are used to decrement data rates. The first type of timer is maintained by sink node. Unless data messages are received within a certain amount of time, the timer causes the last sent interest message to be resent.

Table 5.4 sDIFF Packet Header Format

Name	Description
TYPE	Type of packet (INTR, DATA, DRDY)
SRC	Originating node identifier
UID	Unique packet identifier
LAST	Node identifier for last transmitting node
S_PORT	Port number for originating application
N_NEXT	Number of intended recipients
NEXT	List of intended recipients

The second type of timer is maintained by nodes for each entry in the remote_list cache. Unless a data packet matching the interest entry is received within a maximum interval, the out_rate values in the N[] neighbor array are decremented.

5.3.3 Demultiplexer. This is the same demultiplexer object that is used by the GRP system. It routes incoming messages to the appropriate swarm application via the *AppAgt* interface. Interest messages are routed to the sensor applications where the sensor report rate is changed to the desired value. Data messages are routed to the single swarm application element.

5.3.4 Packet Management. A specialized packet format is used to carry the interest and data messages and the necessary header information to support the gradient routing system. Details of the packet format, along with field descriptions, is given in Table 5.4.

Like the GRP system, the TYPE, SRC, UID, LAST, S_PORT fields are initialized by the *AppAgt* for new packets. The *GradRtr* fills in the NEXT list with the neighbor nodes that have a need for the current DATA packet (see Section 5.3.2.1). The N_NEXT field is used to keep track of the number of neighbors in the NEXT list.

5.3.5 Data Traffic. The scenario used to test the routing protocols includes a single sink (i.e., a single interface to an external agent that has a need for sensor data) and a single source (i.e., a sensor set on a single, randomly chosen node). The sink node sends interest messages beginning at some specified time. Interest messages are resent in one of two cases. First, if data messages are not received within a certain amount of time, the original interest message is resent (see Section 5.3.2.3). Interest messages are also resent when data messages are successfully received. However, these interest messages are sent with higher requested data rates (see Section 5.3.2.2).

Data messages are always sent by the sensor application in response to a need as expressed by the interest messages. New data messages are sent in intervals as specified by the requested data rate. As the requested rate increases, the data updates are sent more frequently.

In order to ensure a fair comparison between the routing protocols, the traffic pattern generated from the sDIFF system is used as the traffic pattern for the GRP and Flooding systems.

5.4 Flood Routing Protocol

The flooding protocol provides a baseline for comparison of the Geographical and Directed Diffusion routing protocols. It is implemented as an optional mode of the GRP system. Figure 5.11 shows a snippet of the simulation script that accomplishes this.

```
for {set i 0} {$i < 4} {incr i} {
  $node_($i) setflood
}
```

Figure 5.11 Flood Node Instantiation

This turns off the GRP router so that any received packet is immediately rebroadcast unless it was previously received.

5.5 Summary

This chapter presents detailed descriptions of the three routing protocols used in the swarm-based sensor network performance analysis. The sDiff protocol provides for route reinforcement while maintaining enough redundancy to adapt to network dynamics. These dynamics can be caused by node mobility (the focus of this research) as well as node failure or external influences (e.g. jamming). The GRP protocol maintains no state information but relies solely on position information to improve efficiency. As a result, the GRP protocol is less complex and easier to implement. Finally, the Flood protocol is used to establish a baseline for comparison of the two target protocols.

The next two chapters use these protocol implementation details and the swarm model to provide an analysis of swarm behavior and the performance of these protocols with respect to that behavior. As is shown in Chapter VII, the sDiff protocol system slightly out-performs the GRP system while both the sDiff and GRP systems out-perform the baseline Flood protocol.

VI. Performance Evaluation Measures and Methodology

This chapter describes the measures[1] and evaluation methodology used to characterize swarm behavior with respect to a sensor network system. In order to place network communications in the context of a swarm-based system, several new measures are proposed to provide a quantitative assessment of swarm behavior. These measures are then used to provide a mechanism for evaluating swarm behavior as it pertains to the dynamics of a swarm-based system.

The swarm measures include a globally based measure and several distributed measures. In addition to the measures, a taxonomy for classifying swarm behavior is proposed.

6.1 Swarm Evaluation Measures

The model described in Chapter IV provides the foundation for analysis of swarm behavior in the context of swarm-based sensor networks. The property of cohesion (connectivity) is important to maintaining reliable communications while the property of avoidance (reduction of sensor overlap) is important to sensor efficiency. Both of these measures are based on global information. An additional measure based only on locally observable phenomena is proposed for providing a distributed behavior identification mechanism. The purpose of these measures is to provide a classification mechanism for categorizing swarm systems. The classification mechanism can be used to recognize changes in swarm behavior and thus provide a means of changing the routing methodology in response to swarm network dynamics.

Other research investigations have been done to establish measures of swarm performance. A method using the ideal gas law is proposed in (60). However, this measure is used to measure a different behavior–namely the time it takes to escape an enclosed region. There are two types of measures proposed. The first is a *global* measure that measures *connectivity*. *Connectivity* is a measure of particle separation distance (which is

[1]Measure is defined in this context a quantitative indication of the extent, amount, dimensions, capacity, or size of some attribute of a product or process. This is not to be confused with a metric or an indicator. A metric is a quantitative measure of how much of a given attribute is possessed by a system, component, or process. An indicator is a metric or combination thereof that provides insight into a process, product, or project itself (84).

important to network communications connectivity). This measure focuses on deviation about the ideal value. The second type is a group of measures used by individual particles to determine behavior in a distributed fashion. This group includes neighborhood size, direction measure and three distance measures.

These measures–and the swarm behavior that they measure–are affected by the various swarm parameters as listed in Table 6.1.

Table 6.1 Swarm Parameters

Parameter	Description	
C_{align}	Alignment weight	(Eq. 4.6)
$C_{boundary}$	Boundary weight	(Eq. 4.14)
C_{periph}	Peripheral vision weight	(Eq. 4.16)
C_{wp}	Waypoint weight	(Eq. 4.21)
$C_{repulse}$	Repulsion weight	(Eq. 4.23)
$C_{attract}$	Attraction weight	(Eq. 4.23)
c_{zone}	Comfort zone	(Eq. 4.24)
α_s	Speed factor	(Eq. 4.28)
θ_{max}	Max turn angle	(Fig. 4.9)
R_{max}	Max cohesion radius	(Eq. 4.25)

These parameters can be related to the swarm behaviors described in Table 2.1. Parameters $C_{repulse}$ and C_{align} relate to *separation* and *alignment* respectively. The parameters C_{periph} and d_3 relate to *cohesion*. The parameter θ_{max} is related to swarm particle capabilities (turn angle). Additional parameters that deal with physical capabilities include maximum velocity and acceleration, inertia effects, size, etc. Variation of these parameters is not considered. For sake of brevity and demonstration of the proposed measures, parameter value sets that result with two distinct behaviors are considered. The two behaviors are *flocking* and *swarming* (as described in Section 4.3.2). For the purposes of this work, the parameters for $C_{boundary}$, C_{periph}, C_{wp}, θ_{max} and R_{max} were held constant. The values for these parameters, shown in Equations 6.1 through 6.5, were derived experimentally (see Appendix E).

$$C_{boundary} = 30 \pm 5 \quad (6.1)$$

$$C_{periph} = 1 \quad (6.2)$$

$$C_{wp} = 10 \pm 5 \quad (6.3)$$

$$\theta_{max} = 4^o \pm 2^o \quad (6.4)$$

$$R_{max} = 4 \quad (6.5)$$

6.1.1 Connectivity. *Connectivity* is measured by examining the distance between particles (specifically the distance between nearest neighbors). This measure quantifies the ability of the swarm formation to maintain cohesion. At each time step the distance between each particle and its nearest neighbor is recorded. The maximum value of this *vector* provides the connectivity information. A plot of the *connectivity* for two parameter sets—one indicated by the shaded region, the other by the unshaded region—is shown in Figure 6.1. The ideal separation distance, as noted before, is unity. A tight, steady formation results in a connectivity measure of approximately one while a more loose formation results in a higher measure. The shaded regions indicate *incoherent* behavior. The parti-

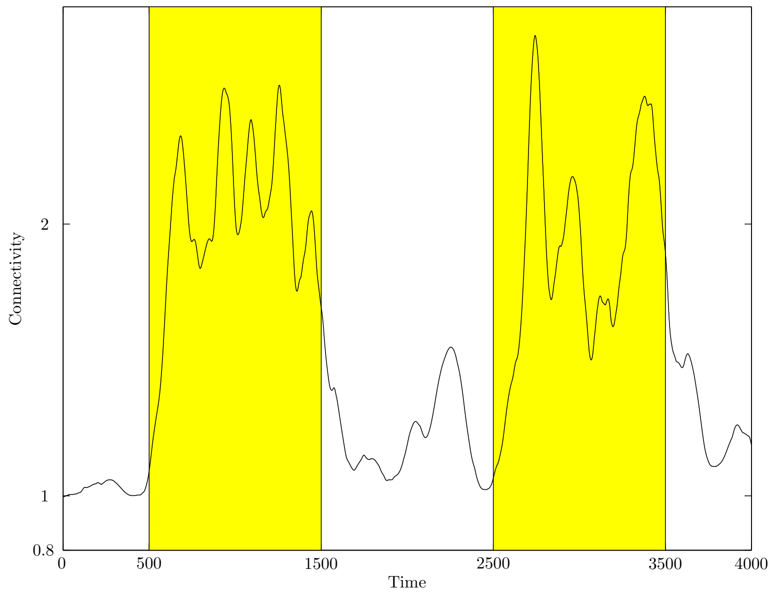


Figure 6.1 Connectivity

cles, while still maintaining a single formation, move in an apparent random sense within the formation. Further, the set of neighbors for each particle changes rapidly. In the unshaded regions, the particles move in a *coherent* formation with little or no change in the neighborhood set. For the purposes of behavior analysis, no boundaries or waypoints are used. This ensures that any variations in behavior are only as a result of the parameters and particle interactions.

Another way to view connectivity uses the time average over a certain period of time. Table 6.2 shows the average connectivity measure for four time segments of three separate and independent simulations. The time segments used for analysis are chosen so as to

Table 6.2 Connectivity Measures

Time segment	Seed					
	7184		1919		3618	
0-500	1.08	0.09	1.02	0.03	1.16	0.20
1000-1500	2.15	0.44	2.10	0.37	1.98	0.35
2000-2500	1.52	0.12	1.28	0.19	1.04	0.07
3000-3500	2.03	0.30	1.96	0.42	2.00	0.30

avoid the transition periods (at $t = 500$, $t = 1500$, $t = 2500$, and $t = 3500$). This is shown graphically in Figure 6.2 where the x-axis is the mean value and the y-axis is the standard deviation. It is not expected that the mean and standard deviations are independent of each other. However, the plot illustrates the *separation* of the behaviors (with flock-like behavior in the lower left corner and swarm-like behavior in the upper right corner). The parameter sets for the two behaviors shown in Figure 6.1 are shown in Table 6.3.

Table 6.3 Behavior Parameter Sets

Parameter	Behavior	
	Coherent	*Incoherent*
C_{align}	8 ± 2	0
$C_{repulse}$	8 ± 2	24 ± 6
$C_{attract}$	6 ± 2	12 ± 3
c_{zone}	0.1 ± 0.02	0.9 ± 0.2
α_s	0.008 ± 0.002	0.012 ± 0.003

Intuition provides some understanding as to the significance of these values. For coherent behavior, a strong tendency to align with neighbors and maintain a tight formation

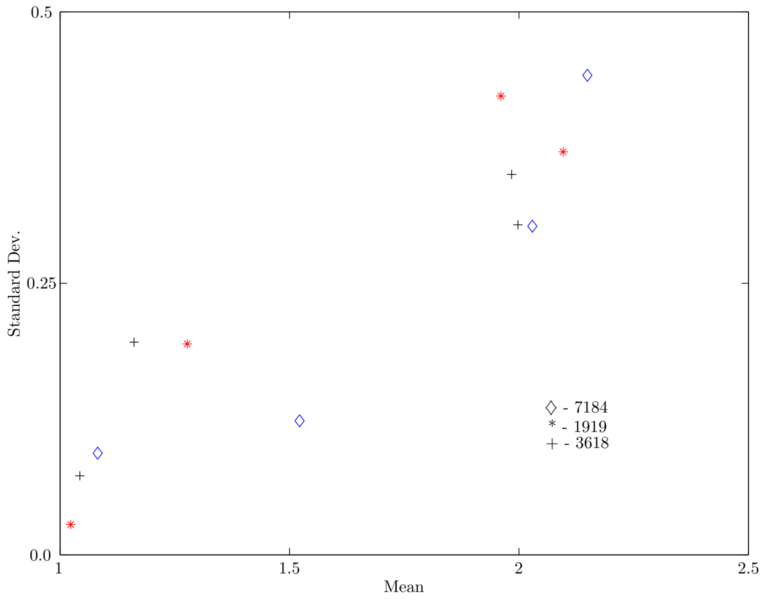
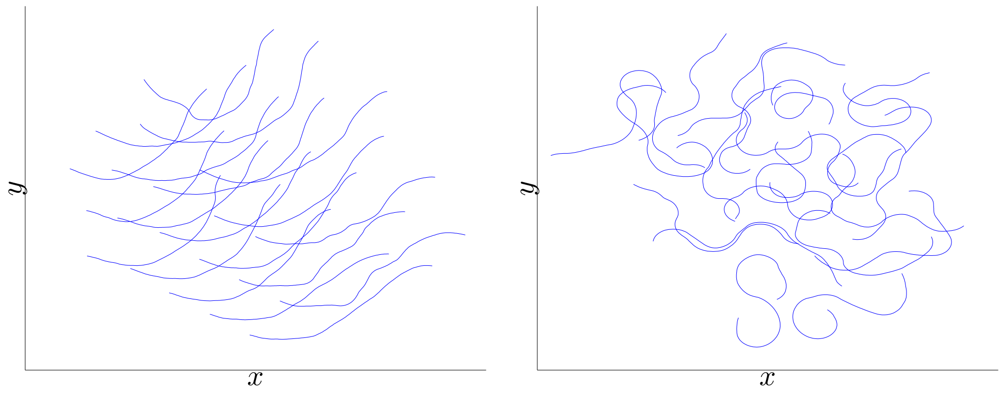


Figure 6.2 Connectivity Variations

is needed. However, incoherent behavior requires a relaxation of the cohesion requirement (via an increase in the comfort zone) and a reduction in the alignment factor.

An example of the two behaviors are shown in Figure 6.3. In this example, there are 20 particles in the formation. In (a), the formation moves consistently with particles matching their direction closely to their neighbors. In (b), the particles remain in a single formation but with little regard for the directions of their neighbors.



(a) Coherent (b) Incoherent

Figure 6.3 Behavior Sample

6.1.2 Distributed Measures. The distributed measures N_i, Ψ_i, D_i, D_i^{min}, and D_i^{max} are computed by each particle independently. They are given in Equations 6.6 through 6.10 (see also Section 4.2.6).

$$\begin{aligned}
&\text{Set size} && N_i = |S_i| && (6.6)\\
&\text{Direction difference} && \Psi_i = \sqrt{\frac{1}{N_i}\sum_{j\,\in\,S_i}(\theta_i - \theta_j)^2} && (6.7)\\
&\text{Distance deviation} && D_i = \sqrt{\frac{1}{N_i}\sum_{j\,\in\,S_i}(d_j - 1)^2} && (6.8)\\
&\text{Minimum distance} && D_i^{min} = \min_{j\,\in\,S_i}(d_j) && (6.9)\\
&\text{Maximum distance} && D_i^{max} = \max_{j\,\in\,S_i}(d_j) && (6.10)
\end{aligned}$$

The formal names of these measures are shown next to each equation. The subscript i indicates that these measures are computed for each particle $\boldsymbol{p}_i$ separately. The set size measure is the same N_i as defined in Section 4.1. It should also be noted that the direction difference of $\theta_i - \theta_j$ in Equation 6.7 adjusted so that the result is in the range of $[-180^o, 180^o]$.

An illustration of these measures for a sample swarm simulation is shown in Figure 6.4. The set size measure shown is the average for the entire swarm at each time step. For the other measures, a windowed average for each swarm member is used to smooth the plots. This provides a better visualization of the swarm dynamics. Clearly, there are variations in all the measures with respect to swarm behavior. The ratios of the mean values for each behavior are listed in Table 6.4. The measure with the largest variation is

Table 6.4 Distributed Measure Variations

Measure	Ratio
N_i	0.98
Ψ_i	7.37
D_i	2.00
D_i^{min}	1.23
D_i^{max}	1.40

the direction difference measure, Ψ_i by a factor of almost 4. This measure is analyzed in greater detail in the next section.

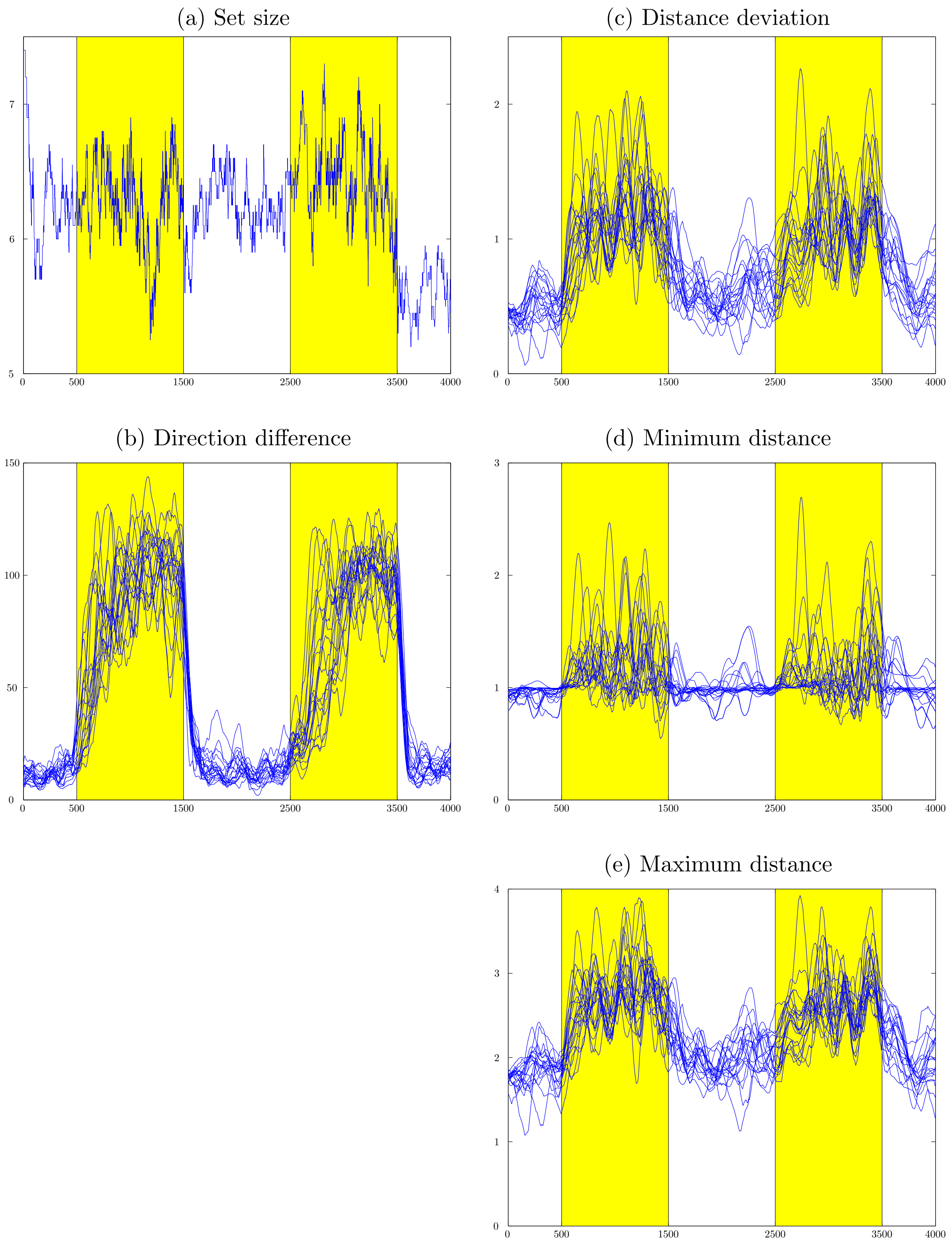


Figure 6.4 Distributed Measures

A challenge in modelling swarms and categorizing swarm behavior is finding a suitable method that does not require a visual evaluation of the swarm as it is simulated. The Behavior Identification Mechanism (BIM) accomplishes the task of behavior identification using a computational method. The measure Ψ_i is described in greater detail.

6.2.1 Behavior Identification Measure. Intuitively this measure is applicable since it captures the essence of the two different behaviors: coherent behavior (CB) and incoherent behavior (IB). For CB, the differences between the directions of a particle and its neighbors should be small while the direction differences are expected to be large for IB. This concept is illustrated in Figure 6.5. The plots are taken from the same data set that was used to generate Figure 6.4. The plots for CB and IB are snapshots at times $t = 2200$ and $t = 3400$ respectively. For sake of discussion, the particles are numbered from 0 to

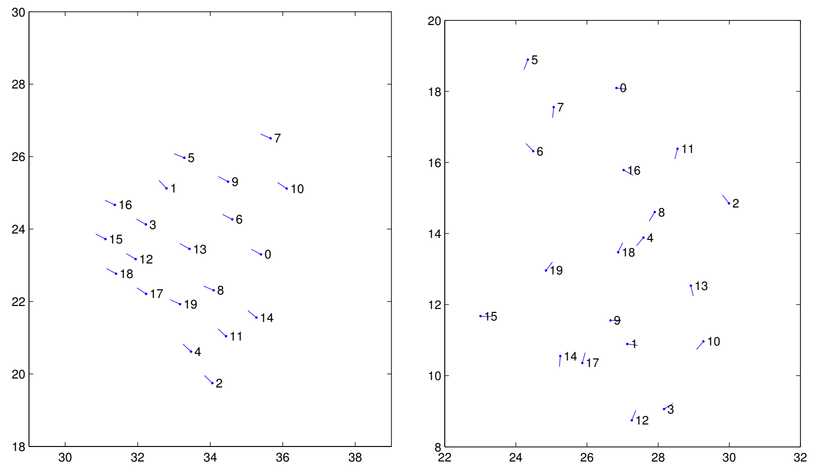
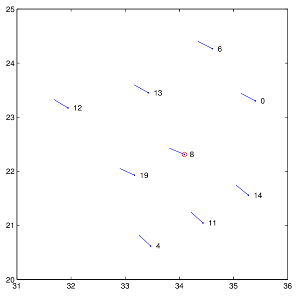
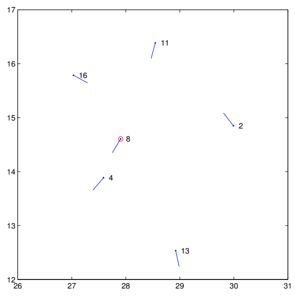

(a) Flock ($t = 2200$) (b) Swarm ($t = 3400$)

Figure 6.5 Formation Direction Vectors

19. The directions are indicated by a vector pointing out the current direction. Note how closely the directions match for the CB and how varied the directions are for the IB.

The plots of Figure 6.5 are enlarged in Figure 6.6 about node 8. Also, only the members of the neighborhood[2] of node 8 are shown.

[2]This neighborhood is the set of *visible* swarm members around node 8 as defined in Section 4.1.1

(a) Flock
$\Psi_8|_{t=2200} = 12.32$

(b) Swarm
$\Psi_8|_{t=3400} = 94.23$

Figure 6.6 Formation Direction Vectors - Enlarged View

The theoretical value for Ψ_i denoted by Ψ (without the subscript) can be determined as follows. First, since the alignment weight (C_{align}) is set to zero it is assumed that the directions are independent of each other and uniformly distributed in the range of $[-180^o, 180^o]$. Let θ_d represent the random variable from which the direction differences are obtained. Then the theoretical value Ψ is simply the square root of the expected value of θ_d^2 which is given by Equation 6.11.

$$\Psi \equiv \sqrt{E(\theta_d^2)} = \sqrt{\int_{-180}^{180} \frac{\theta_d^2}{360}\, d\theta_d} = 103.9^o \tag{6.11}$$

It should be noted that for any random variable X, by definition, $E(X^2) = \mu^2 + \sigma^2$ where μ is the mean of X and σ is the standard deviation of X. The mean of θ_d is zero so the theoretical value Ψ is simply the standard deviation σ.

An illustration of this for the sample data used for Figure 6.4 is shown in Figure 6.7. The data shown is for four time values. These values were chosen randomly within a region that did not include transients due to the change in swarm behaviors. The theoretical value of 103.9^o is indicated by the dashed line and the means of the data sets are indicated by the $\diamond$ symbols. The error bars represent one standard deviation above and below the means.

While an in-depth investigation into the effect of the alignment weight C_{align} on Ψ_i was not done, it is expected that the mean of the random variable θ_d would remain zero while the standard deviation would increase as C_{align} is increased (with all other parameters fixed). For instance, CB is obtained for the parameter set given in Table 6.3.

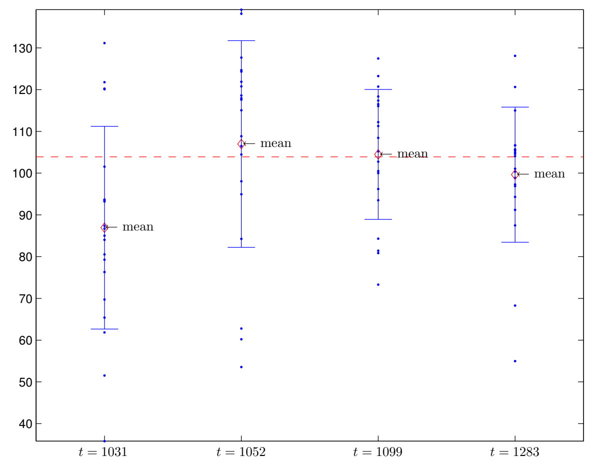


Figure 6.7 Difference Measure Expected Value

If the value for C_{align} is reduced from 8 to 1 very little change is noted. However, as C_{align} is reduced further, the effect becomes significant. The progression of the average and standard deviations for Ψ_i is shown in Table 6.5. The values of Table 6.5 are plotted

Table 6.5 Variation of the Direction Measure

C_{align}	Ψ_i	
	μ	σ
1.0	10.53	5.60
0.5	18.28	14.04
0.4	66.25	21.65
0.3	78.91	27.87
0.2	88.94	23.88
0.1	105.26	18.77
0.0	104.44	18.14

in Figure 6.8. Clearly, the progression is not linear. Further, careful examination of the Ψ_i measure for each value of C_{align} reveals that the progression is not entirely as expected. The phenomenon is discussed further in Section 6.2.3.

6.2.2 Classification Categories.

To determine the efficacy of a particular routing protocol for sensor swarm applications, some method to categorize swarm behavior must be used. A behavior classification mechanism is important since it is conjectured that protocol performance is directly related to the type of swarm behavior.

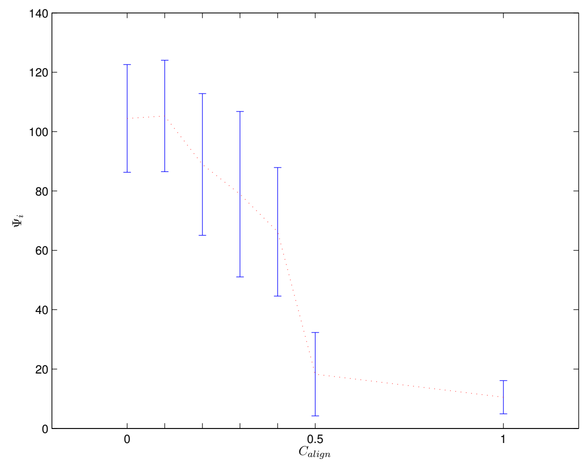


Figure 6.8 Difference Measure Variation

A starting point for suggested categories for classifying sensor swarm behaviors include (type-of) birds, fish, insects, and other animals (66). As noted in Section 4.3, for increased weighting of the alignment rule (parameter C_{align} in Equation 4.6), more rigid formations that maintain a smaller deviation about the ideal minimum distance result. This type of formation is more like a school of fish moving in a non-threatening environment. Reducing C_{align} results in a formation that is more chaotic (greater deviation about the ideal separation distance, as discussed in the previous section) but is more adaptable to environmental conditions (e.g. a swarm of insects).

The taxonomy presented in (37) categorizes multi-agent robotic systems according to communications, computational needs, and other capabilities. However, this taxonomy does not consider the *behavior* of the multi-agent system and the effect of that behavior on communications performance. The concern is with link establishment and duration for wireless communications. As discussed, there are numerous types of swarm formations. A classification scheme is shown in Figure 6.9. The vertical axis represents the scale of the behavior–whether global, i.e. the entire swarm formation, or regional. The lateral axis represents the amount of order in the swarm–ordered like a school of fish or chaotic like a cloud of insects. The depth axis represents the degree of coupling between particles–tightly or loosely coupled in the sense of sharing environment information through some form of communication. Several examples serve to illustrate the innovative classification

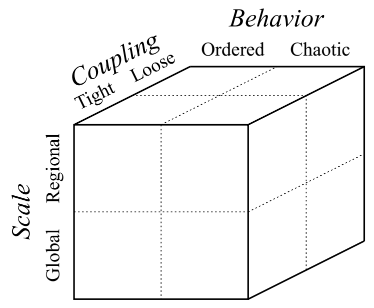


Figure 6.9 Swarm Classification

scheme. A single, large school of fish is an example of a swarm in the *[Global, Ordered, Loose]* class. A colony of ants foraging in widely scattered groups might be categorized as a *[Regional, Chaotic, Tight]* swarm. Finally, a pack of wolves could be classified as a *[Regional, Ordered, Tight]* swarm formation.

Figure 6.9 shows a sharp demarkation between the different regions. However, in reality there is a continuum on which swarm formations may exist. Differing ordered-chaotic behavior can be obtained by varying the parameter C_{align} and the *neighborhood* size. While the other parameters of Table 6.1 are not addressed specifically, their affect can be described from an intuitive standpoint. For instance, decreasing the peripheral weight, C_{periph}, and sight distance, d_{max}, results in a collection of swarms acting almost independently (the *[Global, Ordered, Loose]* class in Figure 6.9).

This classification scheme provides a foundation for evaluating swarms of sensor particles in the context of network communications. Wireless, ad hoc communications protocols employed in these systems can be evaluated and optimized according to specific use in either a static or dynamic sense. Dynamic protocol optimization is important since particle swarms can adapt to the environment and thus fall into a different classification category over time.

6.2.3 Behavior Classification. This section describes the nature of the behavior identification mechanism (BIM) and how it is used to classify swarm behavior. The focus is on the steady state swarm behavior; however, the nature of transitions between behaviors is discussed as well.

Figure 6.4 (b) is repeated in Figure 6.10 for ease of reference. Note that the transitions between states is roughly similar to the output voltage characteristics of a digital circuit. Numerous parameters are used to describe the performance of digital circuits(76, 96).

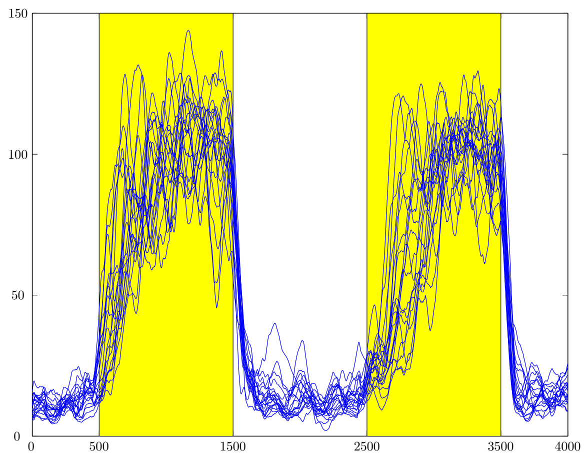


Figure 6.10 Difference Measure Variation

The parameters described include those that are used to characterize the response of a digital circuit to noise as well as the ability of a digital circuit to respond to input signal changes. The former parameters deal with steady state performance while the latter deal with transient performance.

Input and output voltage thresholds and noise margins are used to characterize the response of a digital circuit to changes in the input signal(96). These quantities, illustrated in Figure 6.11, describe how a digital circuit responds to noise on the inputs. The quantities V_{oH} and V_{oL} are the minimum and maximum values respectively for the output of a digital circuit. The quantities V_{iL} and V_{iH} are the maximum and minimum values respectively for which the input is considered low (or high). The noise margins NM_H and NM_L are the differences between the output and input thresholds as shown in Equations 6.12 and 6.13.

$$NM_H = V_{oH} - V_{iH} \quad (6.12)$$

$$NM_L = V_{iL} - V_{oL} \quad (6.13)$$

These measures describe the steady state performance of digital circuits in the presence of noise.

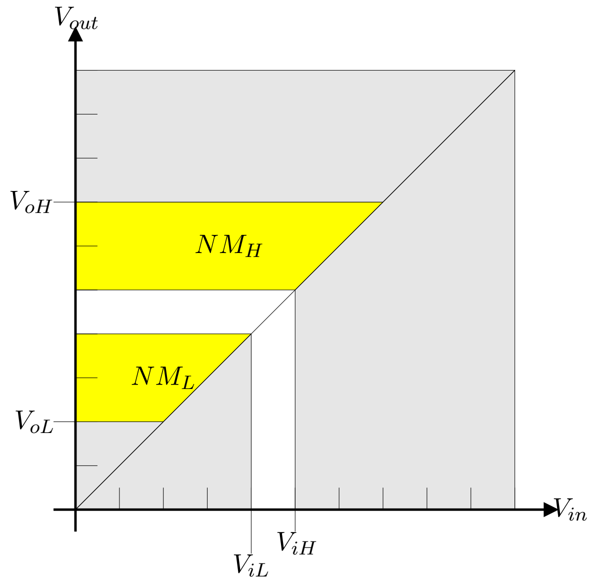


Figure 6.11 Digital Circuit Noise Measures

The measures that describe the output signal transition of a digital circuit include propagation delay and rise and fall times(76). Propagation delay relates the output of the digital circuit to the input and indicates how quickly the circuit responds to changes on the input(s). The rise and fall times measure the rate at which the output voltage transitions from one output level to another. These measures are illustrated in Figure 6.12. There are two propagation delay times–one each for the falling and rising output transitions–t_{pHL} and t_{pLH} measured from the 50% points of each signal. The fall time t_f (rise time t_r) is the time it takes the output signal to transition from 10% to 90% (90% to 10%) of the output signal range.

These measures are adapted for identifying swarm behaviors using the distributed BIM measure. Swarm behavior is limited to two types–CB and IB (as described in Section 4.3.2). The BIM measure is used by swarm members to provide behavior identification. For the purposes of simulation, the BIM provides a mapping between the parameter space $\mathcal{P}$ described in Equation 4.34 in Section 4.3 and the two swarm behaviors. This is illustrated graphically in Figure 6.13. The regions $\mathcal{A}$ and $\mathcal{B}$ represent the parameter space the results in the intended behavior. Let region $\mathcal{A}$ represent flocking behavior and

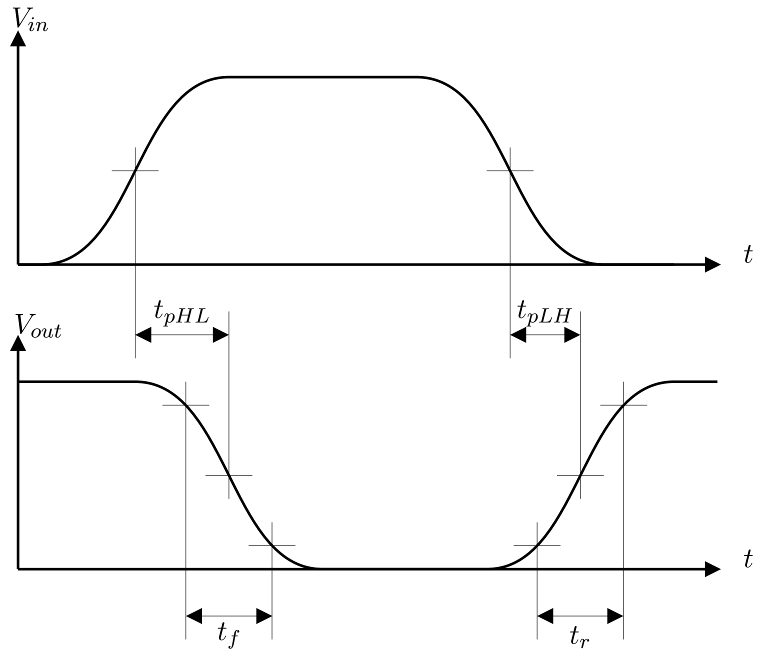


Figure 6.12 Digital Circuit Performance Measures

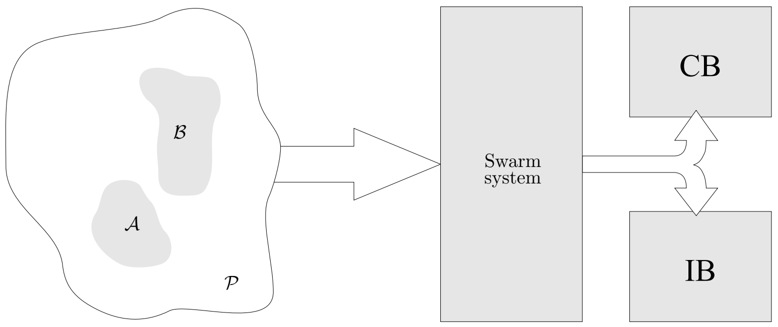


Figure 6.13 Behavior Mapping

region $\mathcal{B}$ represent swarming behavior. In a physical swarm system, the parameters–and mapping–would be implementation dependent.

It should be noted that, even though the regions for CB and IB are shown as continuous and non-overlapping, there is no assumption that this is the case. Additionally, the notional view presented in Figure 6.13 is two dimensional while $\mathcal{P}$ of Equation 4.34 has seven dimensions.

Changes to the BIM values and how that can be used to determine behavior is the emphasis here These changes are related to the changes in parameter values that were used to obtain the two behaviors. The definitions of digital circuit performance measures are adapted to the swarm system.

Plots of the BIM values for two swarm members are shown in Figure 6.14. These plots are generated from the same data used for Figure 6.4. Three plots for each node is

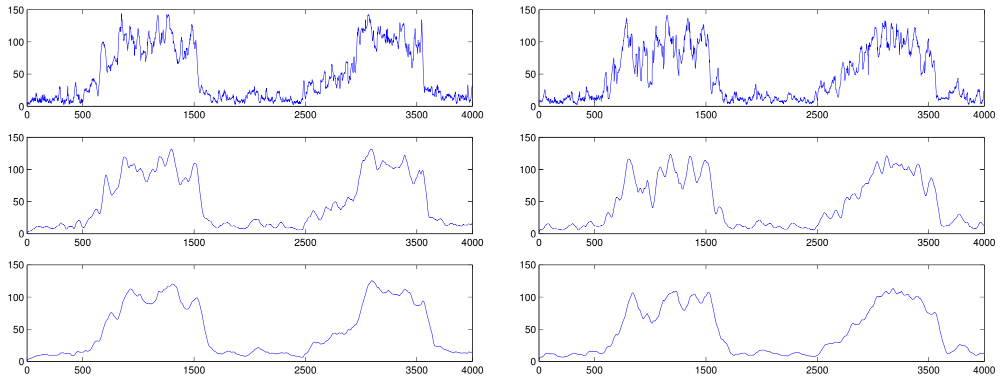

(a) Node 8 (b) Node 13

Figure 6.14 BIM Node Plots

given. The first plot is the raw BIM data while the second and third plots are smoothed versions. These plots are averages of the last 50 and 100 data values respectively. This is shown mathematically in Equation 6.14 where n is the node and L is the length of the smoothing window.

$$BIM_n^{ave}(t) = \frac{1}{L} \sum_{i=t-(L-1)}^{t} BIM_n(i) \tag{6.14}$$

The choice of a value for L involves a tradeoff between smoothness and latency. This is illustrated in Figure 6.15 for node 13. The precise choice of L depends on the application requirements since it is a matter of sample rate. For the analysis here a value of 100 is used. This value is chosen because it provides a much smoother measure while not introducing an excessive amount of latency.

With the context established, the formal definitions of the timing parameters can be given. The parameters similar to t_{pHL} and t_{pLH} are t_{pIC} and t_{pCI} respectively. They are defined as shown in Figure 6.16.

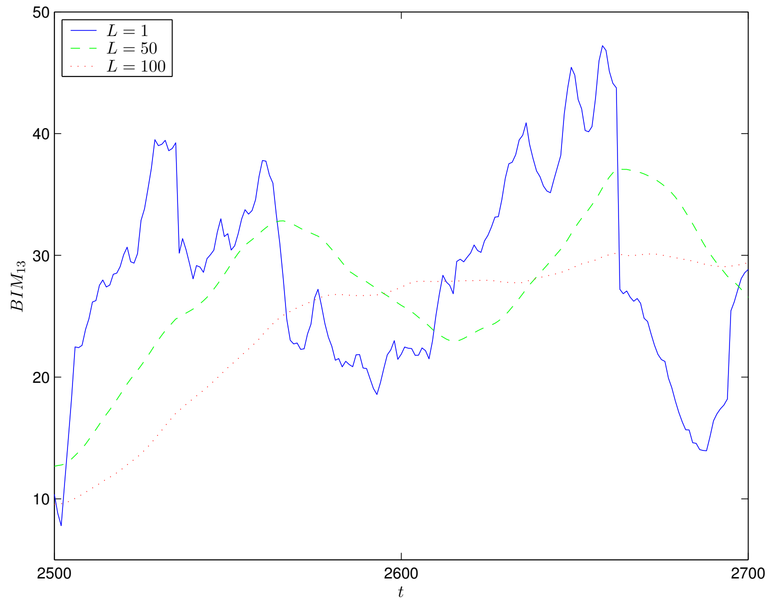


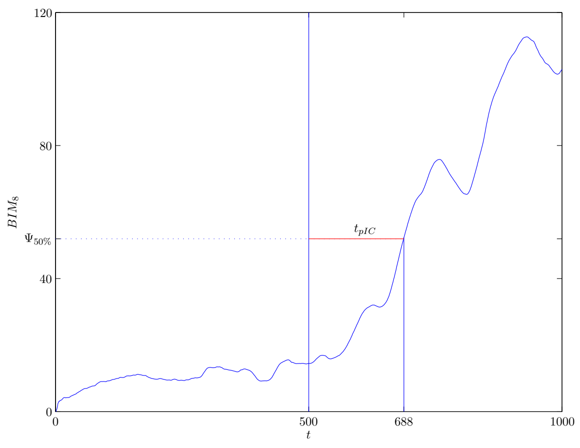
Figure 6.15 Smoothed BIM

The propagation times are defined to be the time from the change of parameters to the time when the smoothed BIM reaches one half of the theoretical average value Ψ (as described in Section 6.2.1).

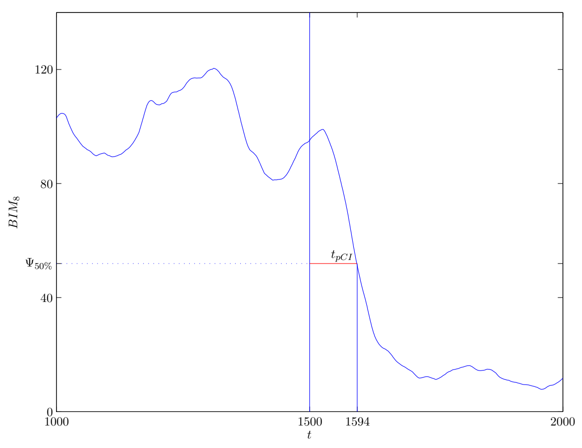
The swarm system equivalent to rise and fall times are the times required to switch from one behavior to another. These quantities are denoted by t_I and t_C are similar to t_r and t_f respectively. The swarm system parameters are defined to be the time it takes the BIM value to go from the 20% point to the 80% point of the theoretical value Ψ. These measures are illustrated in Figure 6.17.

These parameters are only defined *after* the swarm parameters have changed. The need for this aspect of the definition is illustrated in Figure 6.18. The BIM crosses $\Psi_{80\%}$ at $t = t_1$ but this is a false beginning of the transition period since the parameter change does not occur until $t = 1500$.

Noise margins and the associated threshold values do not have an equivalence in swarm systems. In a digital system they have meaning when circuits are cascaded together to form more complex systems. In this sense there is no direct extension of these principles to a swarm system. However, the inputs to the swarm system–the parameters values that



(a) IB to CB ($t_{pIC} = 188$)



(b) CB to IB ($t_{pCI} = 94$)

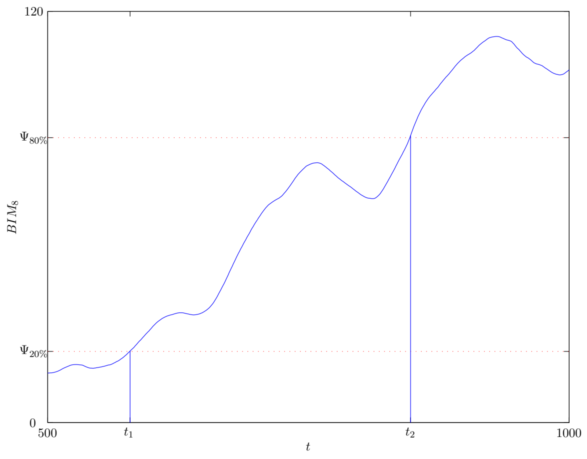
Figure 6.16 Propagation Delay

determine behavior–can vary within certain ranges and still produce a particular behavior. Therefore, the concepts of maximum input low voltage, V_{iL} and minimum input high voltage, V_{iH} can be related to parameter ranges, and the concepts of maximum output low voltage, V_{oL} and minimum output high voltage, V_{oH} can be related to the BIM values. Further, threshold values must be determined for the BIM measure when determining swarm behavior.

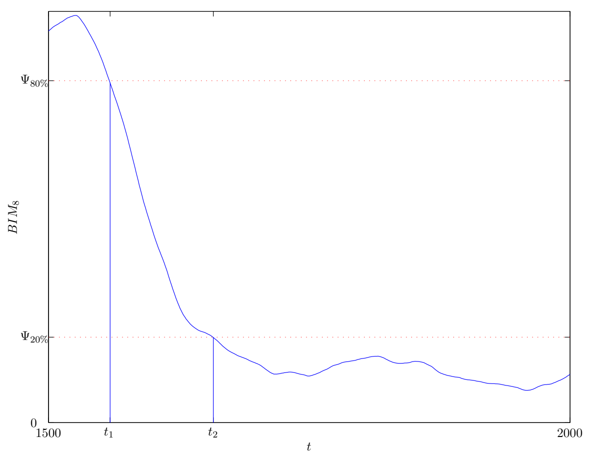
As stated earlier, the dimension of the parameter space is seven. However, in order to accomplish a tractable analysis, it is assumed that several of the parameters are fixed–specifically, the parameters C_{periph} and θ_p. This leaves the parameters C_{align}, $C_{repulse}$, $C_{attract}$, c_{zone}, and α_s. It is further assumed that the transition between behaviors–over *time*–is accomplished by a linear movement in parameter space, i.e., there exists a line that connects the two parameter points. This linear mapping can be expressed parametrically as shown in Equation 6.15

$$p(\lambda) = (p_F - p_S)\lambda + p_S,\ 0 \leq \lambda \leq 1 \tag{6.15}$$

where λ is the parameter of variation and p_F and p_S are generic representations of a swarm parameter for the flock and swarm values respectively.



(a) CB to IB, $t_I = t_2 - t_1$



(b) IB to CB, $t_C = t_2 - t_1$

Figure 6.17 Transition Time

The definitions for output and input voltage thresholds must be stretched significantly in order to adapt them to swarm systems. The parallel to the input voltage thresholds (V_{iL} and V_{iH}) are the parameters t_{HL} and t_{LH} determined experimentally using Equation 6.15. The parallel to the output voltage thresholds (V_{oL} and V_{oH}) is the *average* BIM value for a swarm in steady state. A notional view of these concepts is shown in Figure 6.19. The shapes of these curves resemble experimentally obtained results. These results are presented in Appendix G. The center, darker curve in each plot represents the average steady state value of the BIM measure. The lighter curves denote one standard deviation. The BIM threshold equivalents to V_{oL} and V_{oH} are chosen to be one half of the theoretical BIM value $\Psi_{50\%}$ (i.e., the same for each). The equivalents to V_{iL} and V_{iH} are denoted by the *incoherent behavior (IB) to coherent behavior (CB) threshold*, λ_{IC} and the *CB to IB threshold*, λ_{CI} respectively. Since swarm behavior is a stochastic process, the variation of λ_{IC} and λ_{CI} must be quantized in some fashion. This is accomplished by determining the values of λ where the standard deviation curves cross the $\Psi_{50\%}$ line. These values are denoted by λ^u_{CI} and λ^l_{CI} for the upper and lower variation margins, respectively.

Determining an equivalent to noise margin also requires a significant adaptation. Whereas noise margins in digital circuits are determined by the output voltage of one circuit used to drive the input of another, the "output" of a swarm system is not used to control the "input" of another. In a swarm system the input is the parameter set values–in

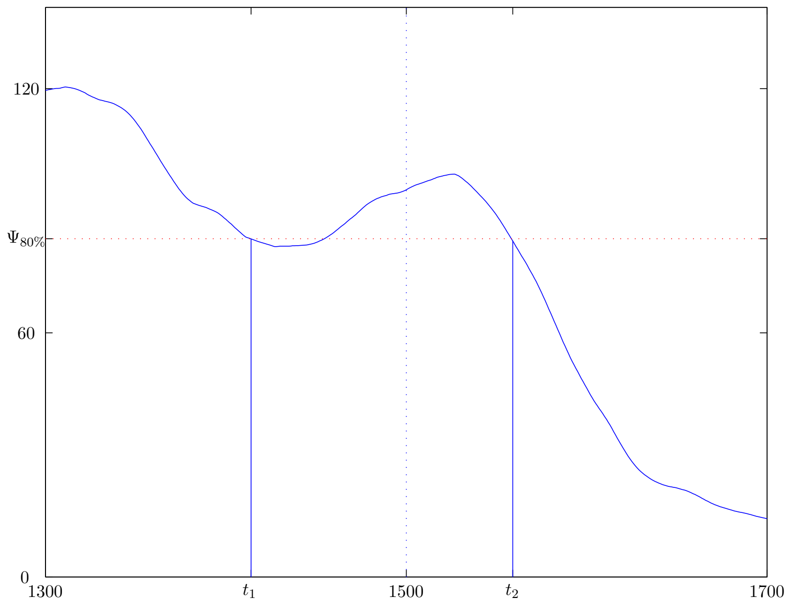
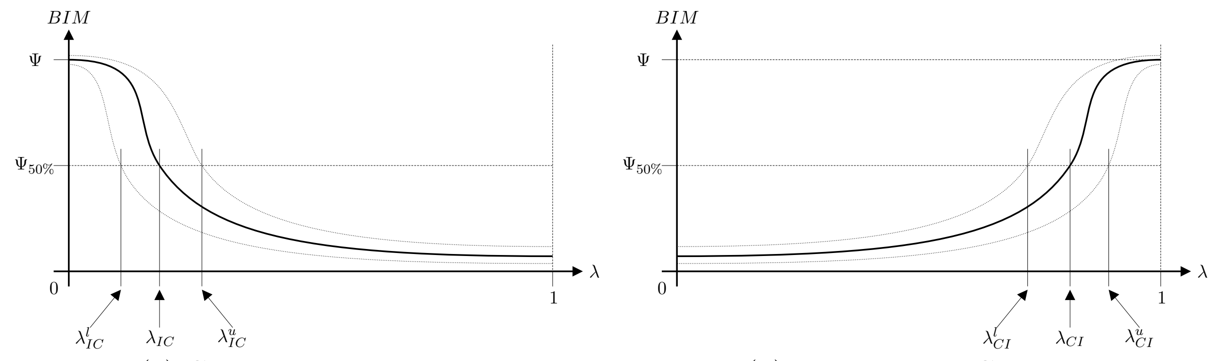


Figure 6.18 False Transition, IB to CB

this analysis, the set of values is determined by λ–without units and the output is the BIM–in units of degrees (or potentially radians). To overcome this, the margins are computed by first scaling λ by Ψ and then finding the difference between that result and $\Psi_{50\%}$. Further, the noise margin is found for the worst case scenarios. These concepts for *CB Noise Margin* NM_C and *IB Noise Margin* NM_I are presented formally in Equations 6.16 and 6.17 respectively.



(a) Coherent to Incoherent (b) Incoherent to Coherent

Figure 6.19 Input and Output Threshold Examples

$$NM_C = \Psi_{50\%} - \Psi\lambda^u_{IC} \tag{6.16}$$

$$NM_I = \Psi\lambda^l_{CI} - \Psi_{50\%} \tag{6.17}$$

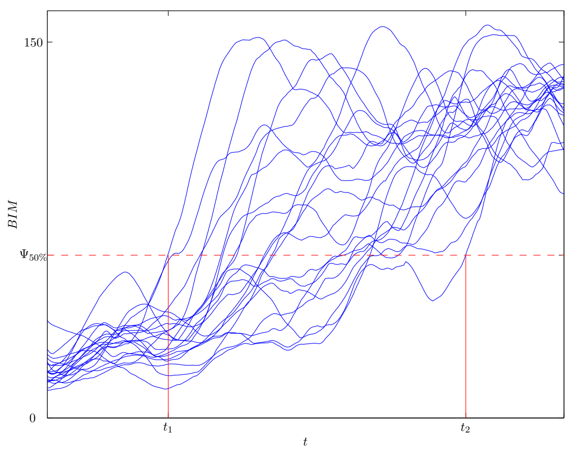
This completes the development of the swarm system measures. The swarm system parameters and their digital system equivalents are summarized in Table 6.6.

Table 6.6 Digital and Swarm System Performance Measures

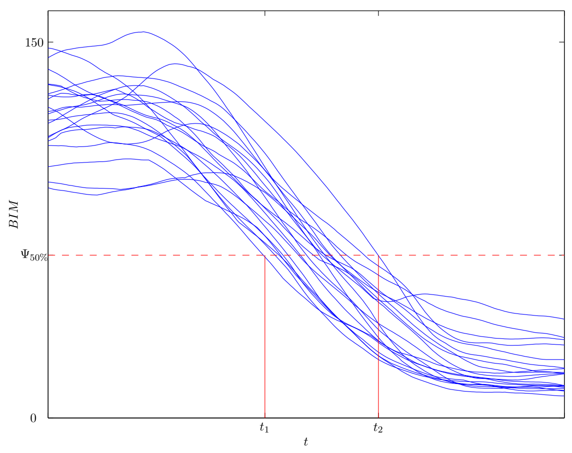
Digital circuit	Swarm system	Description
t_{pHL}	t_{pIC}	Delay time for change to Coherent Behavior (CB)
t_{pLH}	t_{pCI}	Delay time for change to Incoherent Behavior (IB)
t_r	t_I	Time for transition to IB
t_f	t_C	Time for transition to CB
V_{iL}	λ_{IC}	IB transition input threshold
V_{iH}	λ_{CI}	CB transition input threshold
N/A	λ^l_{IC}, λ^u_{IC}	IB lower and upper threshold margins
N/A	λ^l_{CI}, λ^u_{CI}	CB lower and upper threshold margins
V_{oL}, V_{oH}	$\Psi_{50\%}$	CB and IB behavior threshold
NM_L	NM_C	CB noise margin
NM_H	NM_I	IB noise margin

These measures provide a performance analysis mechanism that is individual-based. One other measure is proposed that measures an aspect of the swarm on a global scale. For swarm applications it may be important to be able to characterize the amount of time that is needed to transition the entire swarm from one behavior to another. This measure, which has no parallel in digital circuits, is determined by measuring the amount of time between the time the first particle crosses the $\Psi_{50\%}$ threshold to the time the last particle crosses the same line. This is illustrated in Figure 6.20. The measures, Δt_I and Δt_C, are shown in Figure 6.20 for the same swarm data used for Figure 6.4.

These measures are used in subsequent analysis to characterize the behavior of swarm systems. The behavior characterization is used to evaluate communications protocols for a swarm-based sensor network. In order to provide a concise analysis, network performance measures are used to evaluate the communications protocols with respect to the swarm behaviors. These measures are briefly described in the next section.



$\Delta t_I = t_2 - t_1$
Coherent to Incoherent
(a)



$\Delta t_C = t_2 - t_1$
Incoherent to Coherent
(b)

Figure 6.20 Behavior Transition

6.3 Network Performance Measures

Network performance measures are used to measure the efficiency and effectiveness of a network system. Three measures are used in (15): packet delivery ratio, routing overhead, and path optimality. Packet delivery ratio and path optimality measure effectiveness while routing overhead is a measure of efficiency. As (15) states, packet delivery ratio is important since it describes the loss rate which in turn affects the maximum throughput that the network can support. Routing overhead is important since it gives an indication of how well the protocol performs in congested or low-bandwidth networks. It can be measured in two ways: the number of overhead packets and the number of overhead bytes. Since bytes correlate to transmission rates directly, using bytes is a more accurate portrayal of the actual overhead associated with a routing protocol.

Three measures are used for swarm based sensor networks to measure network performance. The first, *delivery ratio* D_b, is used to measure protocol effectiveness. In this research effectiveness is limited to measuring the amount of *data* that propagates through the network. Therefore, control messages (RQST or INTR packets, see Sections 5.2 or 5.3 respectively) are not included. The other two measures quantify protocol efficiency. *Routing overhead* R_b, encompassing both control packets and data packets, measures the amount of overhead required for each protocol. The other efficiency measure, *data throughput efficiency* E_d, measures the number of data bytes b_{data} received by all nodes for each successfully received data packet p_{data} and is scaled by the data packet size B. These measures are summarized formally in Equations 6.18, 6.19 and 6.20.

$$D_p \equiv \frac{p_{success}}{p_{success} + p_{dropped}} \tag{6.18}$$

$$R_b \equiv \frac{b_{data}}{b_{data} + b_{overhead}} \tag{6.19}$$

$$E_d \equiv \frac{b_{data}}{B p_{data}} \tag{6.20}$$

The symbol p in Equation 6.18 is used to indicate that the D_p measure is based on packets while the symbol b in Equation 6.19 is used to emphasize that the R_b measure is based on bytes as opposed to packets. The symbol B in Equation 6.20 specifies the number of bytes in a data packet. This is a user specified quantity. The value of D_p is limited to the closed interval $[0, 1]$ with the optimum value being unity. The value of R_b is limited to the half open interval $[0, 1)$ with the optimum value approaching unity. This result is due to the fact that, for all three protocols described in Chapter V, the overhead $b_{overhead}$ can never be zero. The E_d measure, unlike the other two measures, indicates improved performance for *smaller* values since larger values indicate more extraneous data packets being sent by all nodes in order to propagate a single data packet. The ideal value (a minimum) for E_d is dependent on the number of hops between the source and sink. As long as comparisons between protocols are made on the same network, the results are meaningful.

6.4 Summary

Measures provide a method for objectively evaluating a system. The swarm system *connectivity* measure allows a direct link between swarm behavior and network communications characteristics. However, its usefulness is limited to a simulation analysis or post-scenario analysis since it requires global knowledge of the swarm.

The distributed measures of *set size*, *direction difference*, *distance deviation*, and *minimum* and *maximum distance measures* are computed by each particle independently.

Therefore, they can be used as a method for *self* behavior identification during swarm scenarios (whether in simulation or in actual execution). As was shown in Section 6.1.2, the *direction difference* measure Ψ_i provides a good behavior identification method since it manifests the largest difference for the two swarming behaviors of Coherent and Incoherent.

These swarm measures can be used in order to identify behaviors that can affect network communications. In order to evaluate the performance of a network communications protocol in swarming applications, three measures are proposed. The *delivery ratio* measures the effectiveness of a protocol while the *routing overhead* and *throughput efficiency* measure the efficiency of a protocol. These swarm and communications measures are used to evaluate the sensor swarm network system described in Chapters IV and V.

VII. Swarm Sensor Network Evaluation

Swarm and network performance testing analyses are presented in order to provide a quantitative understanding and evaluation methodology for swarm based sensor network applications. Additionally, an innovative testing methodology for the three routing protocols in a swarm sensor network is given. The foundations of swarm behavior and network communication protocols are established by Chapters IV and V respectively. The measures used to evaluate the swarm-based sensor network system are described in Chapter VI. This chapter describes the experiments and testing used to investigate swarm behavior and network communications issues—specifically issues for routing protocols—in the context of a swarm-based network as well as the results.

Testing is accomplished in two phases. First, the swarm simulation is tested to determine the characteristics of swarm behavior with respect to the swarm parameters. These characteristics are defined in Chapter VI. The second phase tests the routing protocols in a swarming simulation. Multiple scenarios are used to provide a statistical sampling.

7.1 Swarm Testing

It is conjectured that the behavior of swarms with respect to a given set of parameters (i.e., in steady state behavior) is strict-sense stationary (110). The behavior, regardless of the initial *starting point*, has the same statistics, and the statistics remain constant over time. This section details the tests used to investigate this conjecture. The statistics analyzed include the transition times, Δt_I and Δt_C. Additionally, statistics for neighborhood size are developed for the asymptotic analysis of movement closure as introduced in Section 4.3.3.

The swarm simulations were carried out on three different computing platforms running either Windows 2000^{TM} or Windows XP^{TM} (Microsoft Corp.). The hardware and software of these machines is summarized in Table 7.1. Swarm simulation sizes range from 20 to 100 member particles. A typical swarm simulation of 20 nodes for 8000 time units requires approximately 39 sec. on computer C3.

Table 7.1 Simulation computing hardware

Computer	C1	C2	C3
Model	Dell Dimension T450	Dell Inspiron 7500	HP Pavilion 743
Processor	Pentium 3	Pentium 3	Pentium 4
Speed	450MHz	750MHz	2.40GHz
Memory	384MB	512MB	384MB
OS ver.	2000	2000	XP

7.1.1 Swarm behavior process. In order to determine whether the swarm simulation is a strict-sense stationary process, samples of the transition duration measures Δt_I and Δt_C are tested. Of the swarm measures described in Section 6.2.3, the focus of this analysis is limited to these two measures since group behavior is what determines the states as described in Figure 4.25 and Table 4.7. Transitions between these states are important when considering applications for swarm-based sensor systems.

A sample consists of one simulation run with a different initial random number seed but with the same parameter specification. The parameter specification is given in Table 7.2. The parameter set described in Table 7.2 is repeated for $i = 0, 1, \ldots, 3$, and

Table 7.2 Sample Simulation Parameter Specification

Time	$C_{repulse}$	C_{align}	$C_{attract}$	α_s	c_{zone}
$1500i$	8.0	8.0	6.0	0.008	0.1
$1500i + 500$	24.0	0.0	12.0	0.012	0.9

the simulation stops after 6000 time steps. This results in a 'waveform' of the Behavior Identification Measure (BIM) with four transitions from coherent to incoherent behavior and four transitions from incoherent to coherent behavior.

7.1.2 Statistical Characterization. Three sets of data, each with 100 samples, are used for a statistical analysis and characterization. Figure 7.1 shows the average transition duration along with the standard deviation for the three sample sets. Clearly the rising edge data (with a mean of approximately 100 time units) has a different distribution than the falling edge data (with a mean of approximately 300 time units). The question that must be answered is whether the difference is simply in the parameters of the distribution or are there differences in the types of distributions, i.e., are the distributions uniform,

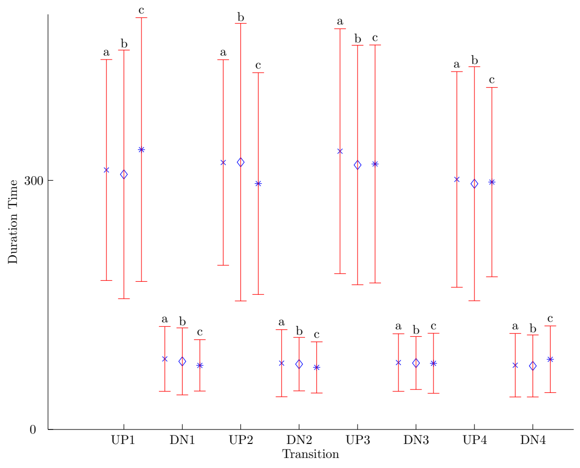


Figure 7.1 Transition Duration

normal, etc. Analysis of the sample data indicates that the distributions have the form of a Rayleigh Distribution (112). The Rayleigh PDF $p(X)$ and Expected Value $E(X)$ are given in Equations 7.1 and 7.2

$$p(X) = \frac{X}{\beta^2} \exp\left[-\frac{1}{2}\left(\frac{X}{\beta}\right)^2\right] \tag{7.1}$$

$$E(X) = \beta\sqrt{\frac{\pi}{2}} \tag{7.2}$$

where $\beta > 0$ is a parameter of the distribution and the random variable $X \geq 0$.

Histograms of the upward and downward transitions for the composite of the three sample data sets are shown in Figure 7.2. The upper plot is for the rising transitions and the lower plot is for the falling transitions. In addition to the histograms, the Rayleigh Probability Density Function (PDF) is plotted using the calculated mean for each transition data set. The horizontal axis of each plot is the transition duration (in time units) and the vertical axis is the frequency with which the transition duration occurs. The histograms and PDFs are both scaled so that the area under each curve sums to unity. Further, the upward and downward transition frequency axes of each plot are scaled by 1000 and 100 respectively.

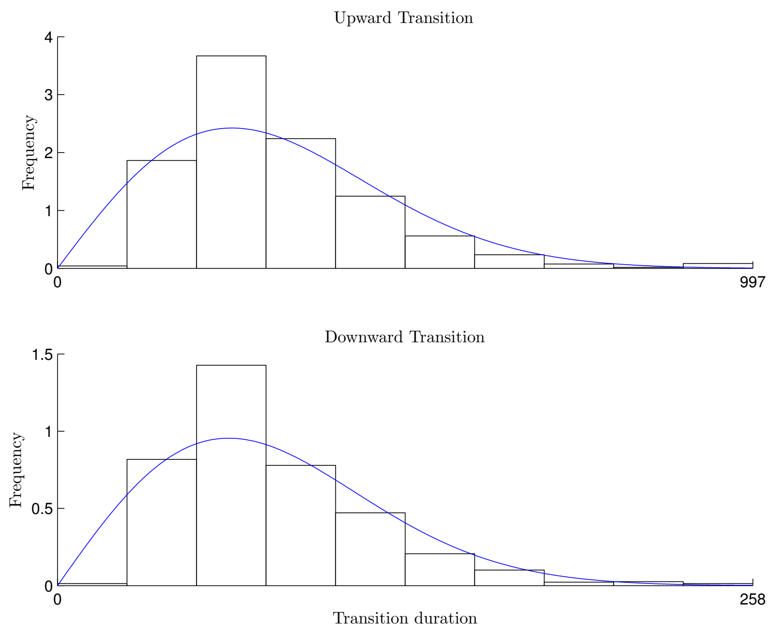


Figure 7.2 Transition Histograms

In answer to the question raised above, it is assumed that the sample data is derived from the same underlying distribution (namely, a Rayleigh distribution). Further, it is assumed that the β parameters for each transition distribution are as shown in Table 7.3. These assumptions are put to the test in the next section.

Table 7.3 Transition Statistical Parameters

Transition	Mean	β
Coherent to Incoherent, Δt_I	312.7	250.3
Incoherent to Coherent, Δt_C	79.6	63.5

7.1.3 Stationarity. The Kruskal-Wallis (KW) test (79) is used to determine the relationship among the sample sets. This is used to test the *null hypothesis*, H_o, that multiple population distribution functions–corresponding to multiple samples–are identical against the *alternative hypothesis*, H_a, that they differ by *location* (17). In other words, the KW test provides a measure of whether the data sets come from the same underlying random process. The KW test is useful since it requires no assumptions concerning the underlying distribution of the samples. The KW test requires at least three random and independent data sets with at least 5 samples per set. Further, the underlying probability distributions from which the samples are drawn must be continuous. The choice of 100 samples per set was based on a tradeoff between the competing objectives of large sample sets and lengthy simulation time. Each simulation is seeded with a different random number.

The KW test generates an H-value which can be used for hypothesis testing. The hypotheses for the comparison of two or more independent groups are:

H_o: The samples come from *identical* populations

H_a: The samples come from *different* populations

The hypothesis makes no assumptions about the distribution of the populations.

The test statistic for the Kruskal-Wallis test is H. This value is compared to a table of critical values for χ^2 distribution (79) with $K-1$ degrees of freedom where K is the number of sample sets (in this case $K = 3$). If H exceeds the critical value for H at some confidence level (usually 0.05) it means that there is evidence to reject the null hypothesis in favor of the alternative hypothesis.

This is useful when determining whether sample sets come from the same distribution for a certain confidence level. In this case, however, the determination needs to be made as to what the confidence level is for accepting the null hypothesis.

Table 7.4 lists the H values and associated confidence values for each of the transition duration sample sets. These values are obtained as follows. From the χ^2 tables one finds the probability that H exceeds χ^2_α where α is the probability. In other words, $P(H > \chi^2_\alpha) = \alpha$ so then, the confidence values in Table 7.4 are the probabilities for *not* rejecting the null hypothesis–as presented above–for each transition.

Table 7.4 Sample Distribution Measures

Transition	H	Confidence (%)
Coherent to Incoherent, Δt_I	2.4298	29.67
Incoherent to Coherent, Δt_C	0.0994	95.15

The interpretation of Table 7.4 is that the null hypothesis of the coherent to incoherent transition can be accepted with approximately a 30% confidence while the other transition can be accepted with over a 95% confidence. One may conclude that the swarm to flock transition data comes from a single random process and is therefore considered stationary, while the coherent to incoherent transition *may* come from a single random process.

7.1.4 Movement Closure. Several methods to implement *closure* of particle movement are possible. Particle speed could be held constant or simply bounded to some maximum value. The former method is not realistic for physical systems while the latter can result in non-linearity of particle movement. As described in Section 4.3.3, it is shown that swarm particle movement is *closed*, and this is accomplished without resorting to either constant speed or hard limits.

The analysis of movement closure begins with Equation 4.59 (repeated here for convenience) with d_2 replaced by its definition (see Equation 4.24).

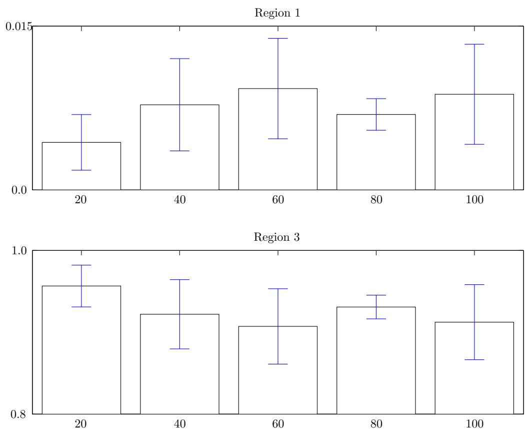
$$|\Delta \boldsymbol{p}_i| \overset{\sim}{\leq} \alpha_s \left[1 + \frac{4}{5}\alpha_1 C_{repulse} + \frac{191}{77}\alpha_3(1 + c_{zone})C_{attract} + C_{align}\right] \tag{7.3}$$

As stated in Section 4.3.3, the bound on particle movement is not a hard upper limit but rather a statistical limit (and hence the symbol $\overset{\sim}{\leq}$). The constants α_s, $C_{repulse}$, $C_{attract}$, and C_{align} in Equation 7.3 are specified parameters that determine behavior. These values are summarized in Table 7.5 for convenience (from Table 6.3). The region proportion parameters α_j, $j = 1, 2, 3, 4$ specify the average proportion of neighbors for each particle in each of the four neighborhood regions (see Table 4.4). The parameter for region R_2 is not shown in Equation 7.3 since the weight is zero. Also, the parameter for region R_4 is not included since there are no particles in that region.

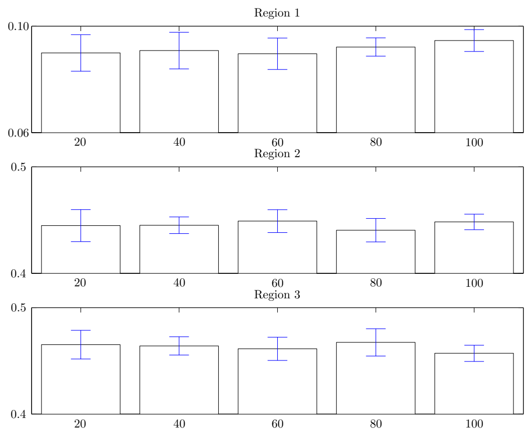
Table 7.5 Behavior Parameter Sets

Parameter	Behavior	
	Coherent	*Incoherent*
C_{align}	8	0
$C_{repulse}$	8	24
$C_{attract}$	6	12
c_{zone}	0.1	0.9
α_s	0.008	0.012

The values for the region parameters are dependent on swarm behavior. Values for these parameters are the result of 10 separate and independent runs of the swarm simulator for each behavior (with swarm size varied from 20 to 100 members). The means for each region are shown in Figure 7.3 for coherent and incoherent behaviors. The plots



(a) Coherent Behavior



(b) Incoherent Behavior

Figure 7.3 Region Proportions

show the mean value with error bars representing one standard deviation for each region as labelled. The variation by swarm size is small for incoherent behavior while the variation for coherent behavior is slightly more. However, since there is no clear trend and the standard deviations overlap, it is assumed that the region proportions do not vary statistically with swarm size. The averages and standard deviations of each parameter over all 10 runs and five swarm sizes are given in Table 7.6. For both behaviors, the proportion of particles in

Table 7.6 Average Region Proportions

Behavior	α_1	α_2	α_3	α_4
Coherent	0.074±0.040	0	0.926±0.040	0
Incoherent	0.091±0.002	0.446±0.003	0.463±0.004	0

region R_4 is zero. Additionally, the proportions of particles in region R_1 for both behaviors is approximately the same.

Using the values of Tables 7.5 and 7.6 the statistical bounds on particle movement for each behavior can be determined. The results are summarized in Table 7.7. The error analysis is made assuming that the parameters α_j, $j = 1, 2, 3$ are statistically independent.

Table 7.7 Calculated Movement Bounds

Behavior	$\lvert\Delta \boldsymbol{p}_i\rvert$	Max
Coherent	0.197±0.006	0.0719
Incoherent	0.347±0.003	0.1298

Additionally, the maximum speed for each behavior for a single simulation is included as an example. Clearly, the calculated bounds are conservative. This is due to the extensive approximations made via the triangle inequality.

7.2 Protocol Testing

This section presents the protocol efficiency and effectiveness measured by protocol overhead and delivery ratio respectively as described in Section 6.3. The results presented are made for *steady state* behavior.

7.2.1 Testing Ground Rules. The physical system modeled consists of vehicles that move according to the swarm simulation. The movement patterns are generated by the simulator and converted to scripts that are run by the *ns-2* simulator. It is assumed that a wireless, radio-frequency physical layer is used with a broadcast, free space propagation model. The transmitter power is set so that the nominal transmission range is 37.7% greater than the ideal separation distance. This value was chosen to ensure that transmission range exceeded the ideal separation distance but did not extend so far as to reach the next hop neighbors. The swarm track data generated by the simulator is scaled so that the ideal separation distance is 60 or 100 meters.

The sDiff protocol does not support multiple senders in its current implementation. As a result single Sensor Application (data source) and single Swarm Application (data sink) nodes are used in each network scenario. The nodes to which the applications are assigned remain fixed for the duration of the network simulation.

Packet sizes are chosen to be 2048 for data packets and 1024 for control packets (INTR or RQST). These values were determined by experimentation (see Appendix E). Larger values caused traffic bottlenecks that resulted in dropped packets because of excessive collisions.

Eleven scenarios are used to simulate the swarm based sensor network–one for coherent behavior and ten for incoherent behavior. Only a single coherent behavior scenario was used since there are no network dynamics (changes in connectivity). The network dynamics for the ten incoherent behavior scenarios are described in Appendix H. The scenarios used provide a wide range of network configurations for exploring the strengths and weaknesses of the three protocols tested.

7.2.2 Performance Results and Analysis. The effectiveness D_p of the communication protocol, as given in Equation 6.18, is a measure of its ability to successfully deliver packets. The measures of effectiveness for the three protocols tested are shown graphically in Figure 7.4. The effectiveness of the network with coherent behavior is unity for all three protocols and so is not shown. The bar charts indicate the effectiveness with the error

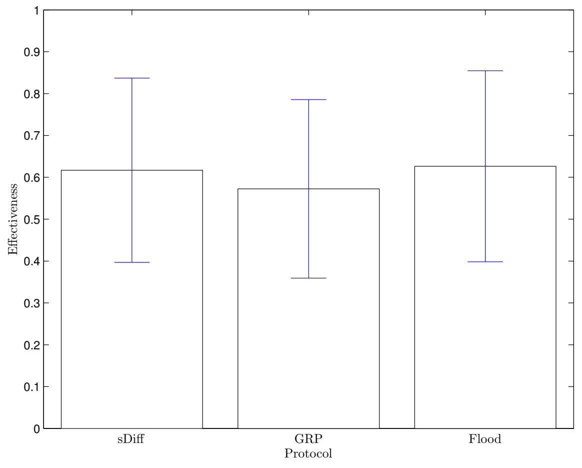


Figure 7.4 Swarm Network Effectiveness

bars indicating one standard deviation. As the plot indicates, no protocol outperforms any other. However, it seems that the GRP protocol is slightly outperformed by the other two.

The control overhead efficiency R_b, as given in Equation 6.19, is a measure of the control overhead required to route data through the network. The values of this measure for each of the protocols is presented in Figure 7.5. The shaded bars for each protocol denote the overhead efficiency for the flock scenario while the unshaded bars are the average of the swarm scenarios. The error bar represents one standard deviation. As with the D_b

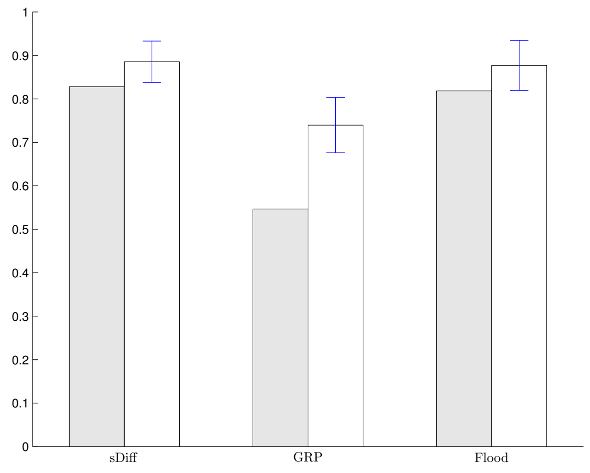


Figure 7.5 Swarm Network Control Efficiency

measure, the GRP protocol is outperformed by the other two protocols but in this case by a clear margin.

The final measure, E_d, measures the data throughput efficiency by determining the number of effective data packets needed to successfully propagate one data packet from source to sink. As with Figure 7.5, the values for this measure are plotted in Figure 7.6. With respect to this measure, it appears that the sDiff and GRP protocols perform about

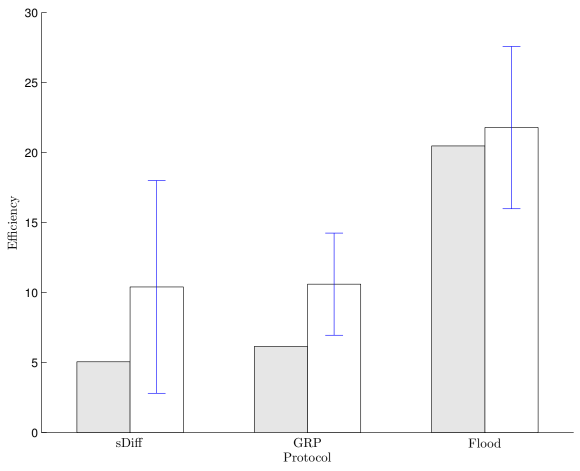


Figure 7.6 Swarm Network Throughput Efficiency

the same. However, unlike the data for the previous two measures, the throughput measure seems to be correlated to the incoherent behavior scenarios. The same plot as Figure 7.6

is shown in Figure 7.7 but with the first four scenarios denoted by the shaded bars and the the last 5 scenarios represented by the unshaded bars. The fifth scenario is not included. As before, the bars represent the mean of the sample data with the error bars denoting one

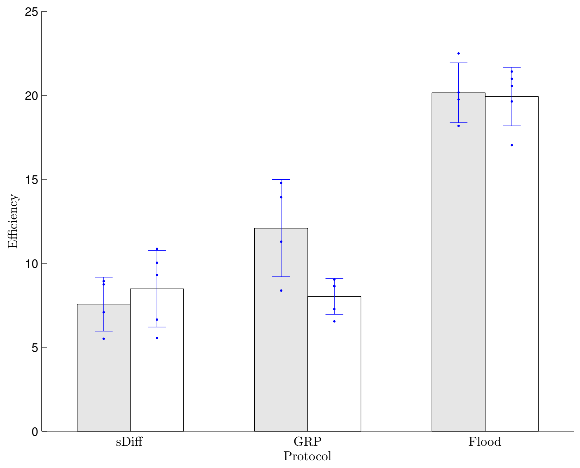


Figure 7.7 Swarm Network Throughput Efficiency by Scenario

standard deviation. The sample data is also included to give insight into the distribution. The difference in the efficiencies is a result of the scenarios used. Specifically, the mean connection duration (see Appendix H) for the first set of scenarios is 79.3% (with a standard deviation of 15.9%), while the mean for the second set is 57.1% (with a standard deviation of 8.3%). The fifth scenario is not included since the small connection duration (29.97%) causes the scenario to generate measures far outside the range of the other scenarios (see the Tables H.2 through H.4 in Appendix H).

While the sDiff and Flood protocols are statistically similar for both sets of scenarios, the GRP protocol differs significantly from one set to the other. It is possible that the GRP protocol could be useful in highly dynamic scenarios.

7.2.3 Network Dynamics and Swarm Behavior. The explanation for this difference is found by inspection of the Behavior Identification Measure (BIM) for each scenario. See Appendix H for more details regarding connectivity analysis using the BIM measure. The means are plotted in Figure 7.8. The error bars represent one standard deviation above and below the means. The dashed line connects the means only to give an indication of the relationship among the points–there is no assumption that the means are the result of any functional relationship among the scenarios.

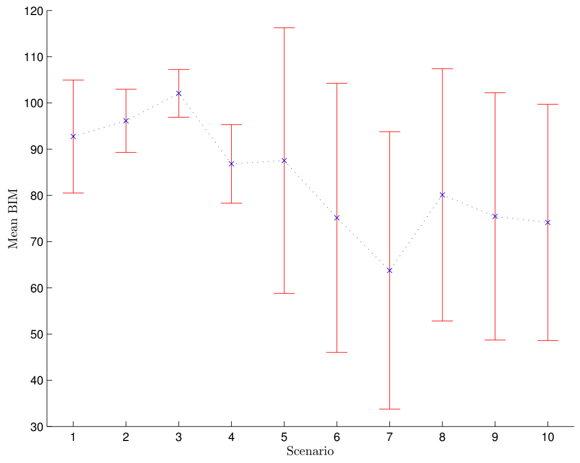


Figure 7.8 Average BIM by Scenario

Figure 7.9 shows the potential correlation between the overall BIM measure versus the data throughput efficiency E_d by the set grouping. The scenarios are partitioned into

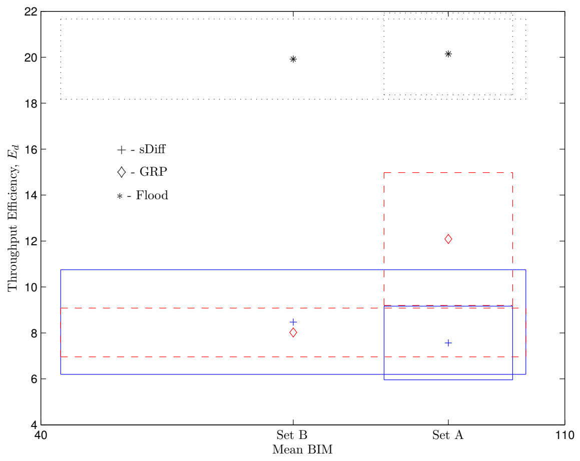


Figure 7.9 Average BIM by Scenario

two sets with scenarios 1 through 4 in Set A and scenarios 6 through 10 in Set B. Scenario 5 is not included for reasons given above. There are six data points: two scenario sets with three data points for each set (for the three protocols, as labelled in the figure). The data points have variation in both the average BIM values and the average throughput efficiencies, E_d. These variations are denoted by rectangular regions around each point.

The variations for the protocols sDiff, GRP, and Flood are denoted by solid, dashed, and dotted lines respectively. For both the sDiff and Flood protocols, the variation regions overlap significantly. However, for the GRP protocol, the regions do not overlap in the efficiency measure. This implies that there is a statistical difference between the two sets with respect to the efficiency of the GRP protocol.

Inspection of the network dynamics for these two sets (see Appendix H) provides the explanation. The GRP protocol does not fair as well for Set A scenarios compared to the sDiff protocol for both sets or the GRP protocol for Set B because of its ability to receive data packets based on direction. This reduces the amount of superfluous data traffic (in Set B) and thus improves the data throughput efficiency.

7.3 Summary

In order to reason about swarming behavior in the context of mobile sensor networks, the swarm model must be characterized. It is shown that the transition duration measures Δt_I and Δt_C are stationary (with approximate confidences of 30% and 95% respectively). Further, the theoretical bounds on movement closure is calculated based on a statistical analysis of neighborhood sizes. The observed maximum position update distances for both behavior is less than these computed bounds.

A new testing methodology is proposed for comparative testing of network communications protocols in the context of a swarm-based mobile network. This methodology is used to compare the three communication protocols sDiff, GRP, and Flood. Results indicate, for the chosen set of parameters, that the sDiff protocol out-performs the simpler GRP system and both sDiff and GRP out-perform the Flood system.

Finally, a link is made between performance of the GRP system and swarm behavior dynamics. The GRP system is better suited for highly dynamic network environments since it is able to propagate data independently of network connections but rather relies solely on directional propagation.

VIII. Conclusion

8.1 Introduction

The research developments presented in this dissertation are original and provide a significant contribution to the general field of sensor networks. It has direct application to the Air Force's Joint Battlespace Infosphere program in the Information Directorate of the Air Force Research Laboratory. These results also have application in the Flight Dynamics Laboratory (AFRL) where autonomous vehicle control research is conducted (25). The Flight Dynamics Laboratory is actively pursuing airborne swarming applications with several companies and universities—they have commissioned Icosystem, Inc. to develop a model for a notional swarm system of up to 110 UAVs based on a pheromone-type process (68). Section 8.2 summarizes the research accomplishments and relates them to the overall research goals. Section 8.3 describes some of the areas where this research can be extended.

Several original concepts contribute to meeting the research goal of developing a swarm based communications modeling process. The swarm algorithm is scalable with respect to behavior and provides a quantitative behavior evaluation methodology. Further, the novel network simulation methodology seamlessly integrates swarm behavior.

8.2 Dissertation Contributions

The specific concepts developed include a novel swarm vision blocking model, a distributed behavior identification methodology, a classification methodology that categorizes swarms by behavior, and communications protocol evaluation in the context of swarm movement. These contributions are now summarized, their originality substantiated, and their specific relationship to the research objectives identified. The objectives, as introduced in Chapter I, are listed in Table 8.1.

8.2.1 Swarm Model Development. The development of the swarm model is predicated on the need for a scalable simulation that maintains *constant* global swarm behavior for a given set of configuration parameters independent of the swarm population size. This

Table 8.1 Research Objectives

Objective: *Swarm Model*
– Investigate state-of-the-art swarming systems/applications
– Implement/improve swarm model
– Investigate swarm scalability issues
Objective: *Swarm Behavior Analysis*
– Develop/analyze behavior measures methodology
– Develop behavior classification methodology
Objective: *Swarm Network Analysis*
– Develop simulation methodology for swarm based sensor network
– Develop comparison protocols
– Develop network evaluation methodology
– Analyze protocol performance in sensor swarm network
– Develop quantitative link between swarm behavior and network characteristics

is achieved through the use of a novel vision blocking model. Additionally, improved global swarm behavior is obtained through the use of a more realistic peripheral vision model.

A significant shortfall in swarm systems is the lack of scalability. The parameter specification is a set of parameters controlling the weights used to update neighbor particles but in a physical system the entire rule set used for position updates must be changed. This is not easily done if it is possible at all. Swarm size can change for several reasons. As applications change, the swarm requirements change. This represents a pre-deployment configuration issue. However, the swarm size can change in the midst of an application also. The environment can change and thus change the nature of the application. Additionally, the swarm size can change due to failures.

As is shown in Section 4.1.1, the vision blocking model allows for global swarm movement behavior that is independent of swarm size. The vision model also ensures that the swarm never loses cohesiveness. Additionally, the vision model was improved to incorporate the concept of peripheral vision. This provides a more realistic model of swarms as they occur in nature and improves the fidelity of the swarm model for simulation purposes.

8.2.2 *Swarm Behavior Analysis.*

A swarm taxonomy is presented based on a novel behavior identification methodology. The taxonomy and classification method provide a mechanism for relating swarm behavior and the associated inter-particle dynamics to the performance of a data communications network used to propagate sensor data throughout the swarm. The behavior identification methodology enables distributed, real-time behavior identification in the swarm.

8.2.2.1 Swarm Classification. A method for swarm behavior classification is needed in order to reason about the relationship between different behaviors and the dynamics associated with transformations from one behavior type to another. The development of the steady-state and transition measures provides a foundation for this type of analysis. The adaptation of digital circuit parameters to swarm systems provides a rich set of measures for developing a classification strategy. These measures are summarized in Table 8.2.

Table 8.2 Swarm System Performance Measures

Swarm system	Description
t_{pIC}	Delay time for change to Coherent Behavior (CB)
t_{pCI}	Delay time for change to Incoherent Behavior (IB)
t_I	Time for transition to IB
t_C	Time for transition to CB
λ_{IC}	IB transition input threshold
λ_{CI}	CB transition input threshold
λ_{IC}^l, λ_{IC}^u	IB lower and upper threshold margins
λ_{CI}^l, λ_{CI}^u	CB lower and upper threshold margins
$\Psi_{50\%}$	CB and IB threshold
NM_C	CB noise margin
NM_I	IB noise margin

While another taxonomy system has been proposed (37), that classification focuses on the technical aspects of the swarm implementation and does not provide a parallel to the natural systems (e.g. flocks, swarms, herds, etc.) which motivate implementations of swarm applications. The natural classification method provides a link to the natural systems and provides insight into how those natural systems (and specifically, their types of behavior) can be used advantageously for sensor swarm applications.

With the Behavior Identification Measure (BIM) changes in behavior can be detected and the sensor network system can adapt to those changes. Along with the BIM, this satisfies all the objectives of the swarm behavior analysis goal.

8.2.2.2 Behavior Identification Measure. The Behavior Identification Measure (BIM), described in Section 6.2.1, provides a mechanism whereby swarm members may identify changes in global behavior. The significant contribution of the method is that it is *distributed*–no global knowledge is needed. As such, it is completely independent of swarm size. The BIM satisfies part of the behavior classification objective as part of the swarm behavior analysis goal.

8.2.3 Swarm Network Analysis. Three communications protocols are compared for performance in the context of a swarm based sensor network. The established Directed Diffusion protocol (58) is compared to the naive, stateless Geographical-based Routing Protocol (GRP) and the baseline Flood Protocol (both developed for this research). The Directed Diffusion implementation used, sDiff, and the GRP implementation fared about equally well with respect to efficiency and effectiveness and much better in all measures than the Flood protocol as is supported by the statistical analyses of Section 7.2. However, the sDiff protocol requires significant node resources (memory and processing) to accomplish the routing tasks while GRP and Flood protocols are simple to implement and require few resources. Taken in this light, the GRP protocol is a viable alternative for swarm based network applications. The development of the GRP system and comparison to the sDiff and Flood systems satisfy the requirements of the protocol development objective under the swarm network analysis goal

A simulation method that tests network performance in a swarm based network did not previously exist. The literature review for this research indicated that the only movement pattern for mobile networking applications used random movement with specified rest times (15, 28, 38, 58). The method used provides a direct link between swarm behavior and the performance of network communication protocols. Further, because of the dynamics of the network, traditional network evaluation measures must be augmented with adapted measures. The data throughput efficiency measure provides a measure of the network performance in the context of network dynamics. These testing and evaluation methodologies together with the GRP development and protocol comparisons satisfy the objectives of the swarm network analysis goal.

8.3 Future Research

Additional work must still be accomplished to investigate potential correlations among the other parameters. The primary focus of this work has been on the alignment rule since this provides the best means by which flock and swarm behaviors may be obtained.

The swarm simulator should be extended for three dimensional swarms. The motivation is two-fold: First, the world is three dimensional and this would make the simulator more realistic especially for airborne applications involving LOCAAS (95) or WASP (35) UAVs. Second, swarm defense methods, as seen in nature, would be significantly different for a three dimensional application.

Other areas of future work should include investigation of the potential parallel between the principles of thermodynamic and particle swarm behavior. Investigation of these principles may lead to a more detailed model of swarm behavior and a more complete understanding of the parameter interactions.

While a cursory investigation of network loading was conducted, a more thorough analysis needs to be done to gain insight into the communications medium use efficiency measures. An analysis needs to be made to determine the maximum theoretical data throughput. One possible maximum can be obtained if one assumes that all nodes are transmitting a the maximum rate. However, this is not realistic since no data is received!

Another area of investigation involves the need for sender/receiver dynamics. This research assumed a single data sink node (Swarm Application) and a single data source node (Sensor Application). The assignment of these tasks remained fixed throughout the simulation. A more realistic scenario should allow the Swarm Application task to migrate through the swarm as required by environmental conditions. An additional augmentation would allow the Sensor Application task to migrate through the swarm in order to follow some fixed location event or phenomenon.

The sDiff protocol does not support *multiple* data producers but the GRP and Flood protocols do. Since a common denominator must be established, scenarios involving multiple data sources was not investigated. However, it would have been useful to see the effectiveness and efficiency measures for each scenario with a single sink and multiple data producers. It is expected that the GRP protocol would fair much better than the Flood protocol since the overhead is much less (and the reduction in overhead would result in less collisions). This conjecture is independent of swarm or flocking behavior.

Appendix A. Particle Swarm Optimization

This appendix provides a more detailed explanation of Particle Swarm Optimization (PSO) techniques and the more general area of Evolutionary Algorithms (EAs).

PSO falls into the broad category of Evolutionary Algorithms(8). These algorithms are useful in finding optimal solutions to highly complex, non-linear single and multiple objective problems(26). More traditional solution techniques, like the Newton method or other hill-climbing algorithms, use local information in the solution space to improve a single solution. The problem with this method is that, in a solution space with multiple local optima and perhaps only a single global optimum, the solution found is highly unlikely to be the global optimum. EAs provide a technique to improve the possibility of finding the globally optimum solution by using a *population* of solutions scattered about the solution *landscape*. This population-based search method can be defined as shown in Equation A.1.

$$P' = m(f, P) \tag{A.1}$$

where P is the initial population in the solution landscape, f is the function for which the globally optimum solution is being search, and m is a population modification mechanism that is used to produce a new population P' based on an optimization of the fitness function f. An example of a fitness function is the generalized Rastrigin function (105) given in Equation A.2.

$$f(\boldsymbol{x}) = \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10) \tag{A.2}$$

where n is the dimensionality of the input domain. This function is known to have a global minimum for $x_i = 0, \forall i \in 1...n$. The three dimensional solution landscape is shown in Figure A.1 (plotted for $\boldsymbol{x} \in [-1.2, 1.2]^2$).

As the plot indicates, the solution space is highly multi-modal. This would cause most optimization functions to settle prematurely on a local optimum.

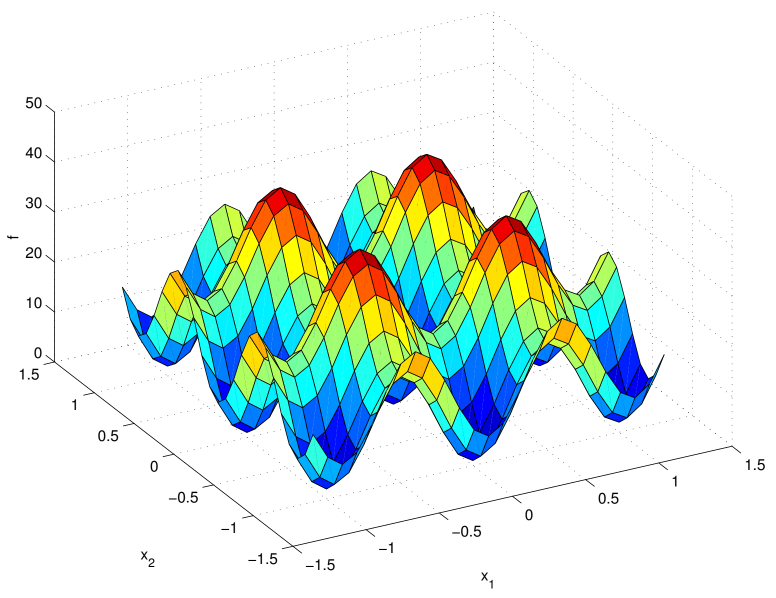


Figure A.1 Solution Landscape

The modification mechanism m in Equation A.1 consists of one or more of the following operations: crossover, mutation, and selection. Different variations of these operators are used to achieve exploration of the fitness landscape as well as exploitation of “good” individuals.

PSO is an algorithmic approach to solving optimization problems that attempts to take advantage of *swarm intelligence*. Swarm intelligence is a property of a system whereby the collective behaviors of simple, homogeneous entities interacting locally with their environment and each other cause coherent functional global patterns to emerge. Examples of such “systems” in the real world are ants and their foraging behavior, termites and their nest-building behavior, and birds in their flocking behavior(69).

The population members in PSO algorithms *move* through the solution landscape in an iterative fashion. Each member moves with a certain *velocity* that is dynamically adjusted according to its *experience* and the experience of its neighbors. Each population member or particle (in a population of size N) in the D-dimensional input space is treated as an infinitely small point. The position and velocity of the i^{th} particle are given by $X_i = [x_{i1}, \ldots, x_{iD}]^T$ and $V_i = [v_{i1}, \ldots, v_{iD}]^T$ respectively. The *fitness* of an individual is a function of its position. The velocities and positions of the swarm members are updated according to Equations A.3 and A.4 at each iteration. The fitness of each particle is given by Equation A.5 where f is a function to be optimized (e.g. find the global minimum or maximum). The previous best position of a particle is given by $P_i = [p_{i1}, \ldots, p_{iD}]^T$. The *global* best solution is denoted by $P_g = [p_{g1}, \ldots, p_{gD}]^T$. P_i and P_g are updated after each iteration also.

$$v'_{id} = v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}),\ \forall i \in 1..N,\ \forall d \in 1..D \tag{A.3}$$

$$x'_{id} = x_{id} + v'_{id},\ \forall i \in 1..N,\ \forall d \in 1..D \tag{A.4}$$

$$F_i = f(X_i),\ \forall i \in 1..N \tag{A.5}$$

In Equation A.3 c_1 and c_2 are positive constants and r_1 and r_2 are random numbers from two independent, identically distributed (usually) random processes. The parameters p_{id} and p_{gd} represent the *experience* of the individual and its neighbors respectively. These values are updated at each iteration according to Equations A.6 and A.7 (for a minimization problem).

$$P'_i = \min(P_i, F_i),\ \forall i \in 1..N \tag{A.6}$$

$$P'_g = \min(P_1, \ldots, P_N) \tag{A.7}$$

Careful incorporation of the swarm behavior rules described in Section 2.3 (see Table 2.1 and Figure 2.5, page 2-6) can result in a robust algorithm that combines exploration and exploitation to find the optimal solution(69).

It should be noted that these algorithms are not guaranteed to find the globally optimum solution in a finite amount of time(69). However, these algorithms have been the subjects of intense research and are continuously being improved in order to converge more quickly with a higher probability on the global optimum.

Appendix B. Wireless Networking Systems

This Appendix provides a summary of several wireless networking systems. The systems described here are the 802.11 system (both a and b), the Bluetooth system, and the Infrared Data Association's IrDA system. These standards address connection requirements, data formats, and protocols for the bottom two layers of the OSI Reference Model (see Section 2.4.1). Additionally, the Global Positioning System (GPS) is described briefly. GPS is a satellite-based location and navigation system and plays an important role in wireless, ad hoc networks.

B.1 802.11

The Institute of Electrical and Electronics Engineers (IEEE) developed the 802.11 standard for wireless local area network (WLAN) architectures(85). The goal of the standards committee was to develop a standard that specified a system that was as similar as possible to the widely accepted 802.3 (Ethernet) standard(119). The major technical challenges are those associated with the greatest benefit of wireless systems: mobility. The standard provides the capabilities to operate a mobile terminal but maintain the traditional level of services found in wired network.

The standard defines two additional systems, the 802.11a and 802.11b. The 802.11a architecture provides data rates up to 54 megabits per second (Mbps) in the Unlicensed National Information Infrastructure (UNII) band (at 5 GHz). The 802.11b architecture provides a slightly slower data rate of 11 Mbps in the Industrial, Scientific, and Medical (ISM) band (at 2.4 GHz).

B.2 Bluetooth

Bluetooth is a short-range wireless networking specification (13). It is being developed by the Bluetooth Special Interest Group, an alliance of vendors consisting of Ericsson, 3Com, Lucent Technologies, Microsoft, Motorola, and Nokia just to name a few. The system was initially developed by researchers at Ericsson and hence the name—the system was named for a tenth-century Nordic King Harald Bluetooth (33). The Bluetooth specification was developed outside of the IEEE standards process. However, the IEEE recognized the need to develop a wireless personal area network and have since incorporated the Bluetooth specification into the 802.15 working group considerations.

The Bluetooth standard specifies the entire protocol stack. Bluetooth is designed to support Personal Area Networks (PAN) at raw data rates up to 1 Mbps over a range of 10 meters. For this reason, Bluetooth is not meant to replace corporate, office or home networks. The application of Bluetooth is limited more to utility types of implementations. This includes wireless headsets for cellular phones and short range interfaces for personal digital assistants (PDA). Bluetooth equipped devices operate in the ISM band at 2.4 GHz.

B.3 IrDA

The Infrared Data Association was established, like the Bluetooth consortium, to develop an open standard for infrared (IR) data communication (129). The specification provides for point-to-point data and control communications (57). IrDA provides high-speed, cordless, line-of-sight data transfer for digital devices—the same devices supported by Bluetooth (i.e. PDAs, laptops, desktop computers, etc.). The IrDA Control specification describes the interface procedures for keyboards (1-way), joysticks (2-way, low latency), etc. It also includes “remotes” for household devices like video cassette recorder/players and televisions.

IrDA was initially designed as a method to replace the proliferation of computer cables(57). However, its utility in providing seamless access through a wireless interface makes it ideal for personal area networking applications.

B.4 Performance Characteristics

The performance capabilities of the various wireless access technologies vary slightly for current or near term systems. The predominant data rates available for wireless data is approximately 1 - 2 Mbps. Plans for next generation systems are in the works to boost these rates nearer to those of currently available wired systems (10 Mbps). Table B.1 lists the performance characteristics of most of the systems above.

Table B.1 Wireless Systems Performance

	Data rate(Mbps)	Range(meters)	Frequency(GHz)
802.11	2	100	2.4
802.11b	11	100	2.4
802.11a	54	TBD	5
802.15 (Bluetooth)	<1	10	2.4
802.15 (high-rate)	20+	TBD	2.4/5
IrDA	4	1	IR

Because of significant overhead in the 802.11b system, the effective data rate is about 6 Mbps. Also, there is a proposal to develop a 16 Mbps standard for IR devices.

B.5 Global Positioning System

The Global Positioning System (GPS) is a worldwide satellite-based radio navigation system(74). The system consists of 24 satellites in six orbital planes operating in circular, 10,900 nautical mile orbits at an inclination of 55 degrees in a 12 hour period. The radio system operates on two frequencies in the L band–L1 is 1575.42 MHz and L2 is 1226.6 MHz– and can be used anywhere near the earth's surface. However, it is line-of-sight dependent and therefore the accuracy of the system suffers in environments where signals get blocked (like in large cities with tall buildings).

GPS receivers can provide longitude and latitude with only three satellites. A fourth satellite is necessary in order to determine the altitude of the receiver. Using even more satellites improves the accuracy which is typically around 15 meters. An enhance GPS system, called differential GPS, uses ground stations to augment the satellites. These systems can achieve accuracies to within a few meters.

Appendix C. OPNET and ns2 Performance

This appendix details the performance differences between the two popular network simulation tools OPNET and *ns-2*.

C.1 Performance Testing

Simulations were executed to test the performance of the OPNET and *ns-2* environments. Test simulations consisted of different sized swarms for different lengths of simulation time. In order to obtain a broad perspective of the simulation programs, tests were run on two different platforms and three different operating systems (OSs). Table C.1 provides the details of the various platforms and OSs. Both simulators were run on the

Table C.1 Test Platforms

	Ultra 10[a]	Inspiron 7500[b]
Hardware	UltraSPARC-IIi	Pentium III
	440MHz	750MHz
	1024MB RAM	512MB RAM
Operating System(s)	SunOS 5.8	Linux[c]
		Windows 2000[d]

[a]Sun Microsystems
[b]Dell Computer Corp.
[c]Mandrake 8.1
[d]Microsoft Corp., Professional version

Ultra 10 under SunOS 5.8. On the Inspiron 7500, OPNET was run under Windows while *ns-2* was run under Linux.

Simulations of five swarm sizes were run–20, 35, 50, 70, and 100 nodes. Each simulation was run for five different durations–20, 30, 40, 50, and 60 seconds. Additionally, in order to provide a statistical sampling, each simulation was run ten times. This data was used to generate the first and second order statistics (average and standard deviation).

The simulation scenario in each case involves a single node sending out one RQST packet. This occurs at simulation time, $t = 15s$. Every node is instantiated with a single sensor application. These applications are started at a simulation time, $t = 10s$ and allowed to produce RESP messages as specified earlier until the end of the simulation. Due to the nature of the application design, RESP messages are not forwarded for transmission by the MAC layer unless a RQST message was received. Therefore, the run-time contribution by the first 15*s* of the simulation is negligible. There are other subtle differences in the run-time performance of the two simulations. These are described in greater detail in the next section.

It should be noted that a significant performance issue was discovered while comparing the run-time performances of the two simulators. Because of differences in the implementation of the 802.11 MAC models, the number of dropped packets due to collisions is slightly different. It was noted that performance is closely related to the number of successfully transmitted packets. Therefore, a delivery ratio measure, as shown in Equation C.1, was used to ensure that the two simulations were processing approximately equal numbers of packets.

$$R = \frac{n_{success}}{n_{dropped} + n_{success}} \tag{C.1}$$

This was done by adjusting the size of the RESP packets. The initial size was set at 4096 bytes[1] for both simulators. With this value the *ns-2* simulator 802.11 MAC model dropped more packets than the OPNET simulation. Once this issue was identified, the packet size for the *ns-2* simulation was reduced until the delivery ratios were approximately equal. Table C.2 lists the delivery ratios for OPNET and *ns-2* (before and after adjustment of packet size).

Table C.2 Packet Delivery Ratio

Swarm Size	OPNET	*ns-2*	
		Before	After
20	96.4	90.2	96.8
35	94.0	82.2	93.9
50	92.5	70.8	90.0
70	91.7	66.7	87.4
100	89.9	69.8	87.0

Additionally, the average inter-arrival time of 2.0s was chosen to further reduce packet collisions. The desire was to ensure that the run-time of the simulators is linear with respect to simulation duration. At an average inter-arrival time of 1.0s, the run-times of both

[1]This was deemed to be a reasonable size for encapsulating a <u>moderate</u> amount of sensor data.

simulators are not linear. However, the same trend with respect to run-time performance and dropped packets was noted (as described in Section C.2.2).

C.2 Results

The assessments of the two simulation environments fall into two categories: Objective (based on an evaluation of the run-time performance) and Subjective (based on ease of use, learning-curve, etc.). The statements made in this section with respect to the subjective issues are based on the authors' experiences. These comments are intended to provide insight into the issues associated with using each simulator and a foundation for choosing a simulator for a particular modeling challenge.

C.2.1 Performance Analysis. This section details the run-time performance of the two simulators. In addition to a presentation of the run-time performance, an analysis is made to develop a predictive mathematical model of the run-time as a function of swarm-size and simulation time. Figures C.1 and C.2 show the run-times for *ns-2* and OPNET

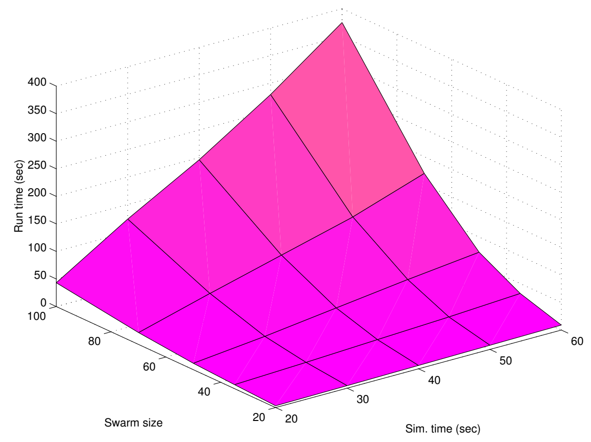


Figure C.1 *ns-2* run-time

on the Sun platform. As these figures show *ns-2* executes in a much shorter time. A cursory analysis of this data indicates that the run-time is linear with respect to simulation time and polynomial with respect to swarm size (for the chosen set of parameters) as is anticipated. These assumptions are used to develop a mathematical model of each of the simulator's run-time performance in the next section. For completeness, an exponential

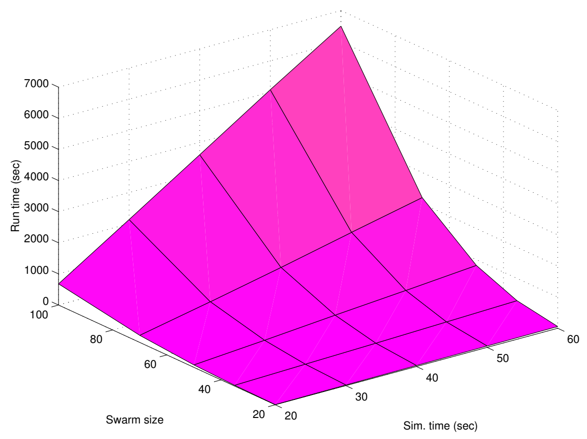


Figure C.2 OPNET run-time

model was investigated for its applicability. However, the values are approximately 2.9 and 2.4 for OPNET and *ns-2* respectively. Therefore, the run-time performance of OPNET is modelled as a cubic polynomial and that of *ns-2* is modelled as a quadratic.

Table C.3 provides the ratios of the run-times for OPNET and *ns-2* for the Pentium III Dell system (for OPNET running under Windows and *ns-2* running under Linux). It should be noted that, for large simulations (consisting of a large number of nodes for long simulations), *ns-2* provides a significant reduction in run-time. Similar data is obtained

Table C.3 Performance Ratio - Dell workstation

Swarm	Simulation Time				
Size	20	30	40	50	60
20	2.73	7.15	7.46	8.16	8.78
35	4.96	9.04	10.9	11.8	11.4
50	8.85	12.4	13.0	13.0	13.6
70	12.3	13.2	14.1	14.9	14.6
100	16.5	17.5	18.7	19.0	18.6

for execution on the Sun workstation. These ratios are shown in Table C.4. Analysis of

Table C.4 Performance Ratio - Sun Workstation

Swarm	Simulation Time				
Size	20	30	40	50	60
20	1.90	4.92	5.41	5.99	6.19
35	3.81	7.23	8.04	8.85	8.88
50	6.80	10.2	10.8	10.8	11.3
70	10.6	12.1	12.9	13.5	12.9
100	15.7	17.2	18.4	18.2	17.3

error propagation for the results in Tables C.3 and C.4 reveal that the deviations for all numbers are less than 0.06 and 0.4 for the Dell and Sun workstations respectively.

The potential explanation for large performance difference: OPNET has more overhead during simulation due to statistics collection. The precise nature of this overhead is unknown due to the fact that OPNET simulator source code is unavailable for analysis. While no statistics were selected for collection during run-time, the overhead associated with this capability might still result in some delay.

C.2.2 Mathematical Analysis. The mathematical analysis of the run-time data is presented here. It should be noted that these results, as they relate to the performance of OPNET and *ns-2* are applicable only to the particular scenario tested. However, the analysis provides useful insight into the performance aspects of the two simulation environments.

The goal is to develop a mathematical model of the run-time of the simulators with respect to the size of the swarm and the duration of the simulation–as shown in Equation C.2

$$t_{rt} = f(n, t_{st}) = g(n)h(t_{st}) \tag{C.2}$$

where t_{rt} is execution time, n is the swarm size, and t_{st} is the simulation duration. As the equation shows, the assumption is that the contributions to the run-time by the two input parameters are uncorrelated and are the result of the product of two functions g and h, each of a single variable.

After analysis of the OPNET run-time data, the form of the relationship was determined to be cubic with respect to swarm size, n, and linear with respect to simulation duration, t_{st}. This is shown in Equation C.3

$$t_{rt} = (A_{opnet}t_{st} + B_{opnet})p_3(n) \tag{C.3}$$

where $p_3(n)$ is a cubic polynomial in n. The form of the *ns-2* relationship, shown in Equation C.4, is also linear with respect to simulation duration but is quadratic with respect to swarm size.

$$t_{rt} = (A_{ns}t_{st} + B_{ns})p_2(n) + h_{src}(n) \tag{C.4}$$

The additional term, $h_{src}(n)$, in Equation C.4 models the time required to read in the node movement and packet traffic source files. It was determined that $h_{src}(n)$ is linear with respect to n. The values for A and B are given in Table C.5.

Table C.5 Simulation Duration Coefficients

	Sun (Unix)		Dell (Linux)	
	A	B	A	B
OPNET	0.1683	-2.381	0.1072	-1.342
ns-2	1.048	-17.09	0.6711	-9.675

The data presented in Tables C.4 and C.3 was generated by evaluating the expression given in Equation C.5 with experimental data.

$$R_{perf} = \frac{(t_{rt})_{opnet}}{(t_{rt})_{ns}} \tag{C.5}$$

A general formula for performance improvement obtained from using *ns-2* can be determined by using asymptotic analysis on Equation C.5. As the size of the swarm increases (with t_{ts} held constant), the performance improvement is approximately linear. With the swarm size held constant, the performance improvement reaches a constant value. This value is specified by a ratio of the constants A_{opnet} and A_{ns} in Equations C.3 and C.4 respectively. Equations for the asymptotic performance improvement on the Sun and Dell platforms are reflected in Equation C.6.

$$R_{perf} \approx \begin{cases} (0.161)n & - \quad \textit{Sun} \\ (0.160)n & - \quad \textit{Dell} \end{cases} \tag{C.6}$$

Plots of the actual performance ratio data (from Tables C.3 and C.4) and the results of the performance ratio models are shown in Figure C.3. The top row of Figure C.3 shows a surface plot of the actual performance ratios for the Dell and Sun platforms respectively. Surface plots of the mathematical models are shown in the bottom row. Inspection of the figure reveals that there are inaccuracies in the mathematical model for small swarms ($<$ 50 nodes). This is not unexpected since it was assumed that the overhead associated

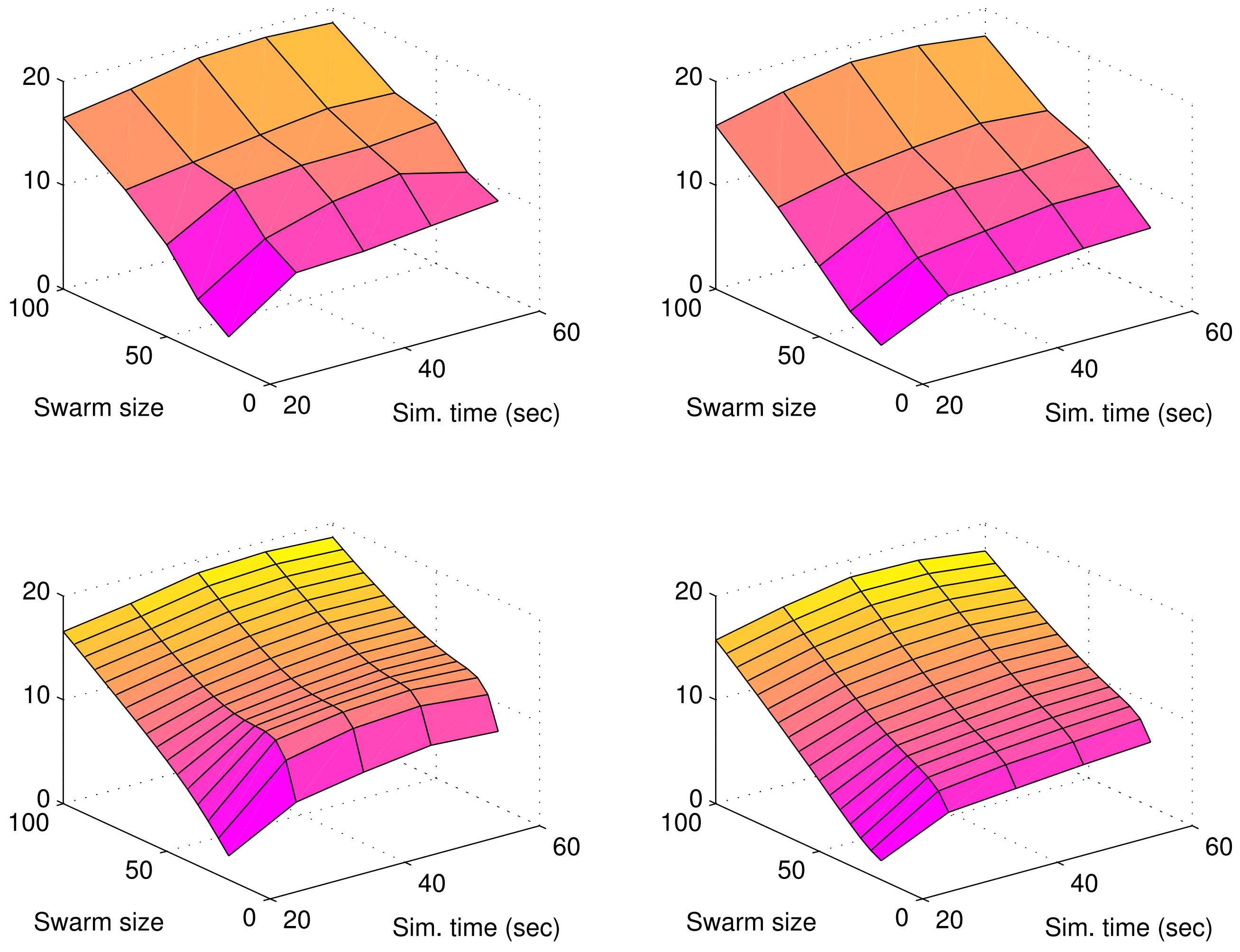


Figure C.3 Actual and predicted performance ratios of OPNET and *ns-2* simulators

with execution of each simulation environment was negligible. For smaller swarms, this assumption probably does not hold.

Not surprisingly, the coefficients in Equation C.6 are approximately equal. This reveals that the operating systems (Linux and MS Windows on the Dell platform) have negligible effect on the performance of the two simulators.

C.2.3 Development Environment. This section details issues associated with the ease or difficulty of model development in each of the simulators. Different development goals determine which package is more efficient or effective.

OPNET, with its rich graphical user interface (GUI), is by far the easier system to learn and use for "out-of-the-box" simulations, i.e., for simulating standard network systems using pre-existing protocols and components (queues, sources, protocols, etc.). In addition to the rich GUI, OPNET offers many analysis tools for automatically collecting and easily visualizing network simulation performance data. Additionally, tutorials (86) are included with the OPNET documentation that can be used to explore the OPNET development capabilities.

The *ns-2* simulator supports numerous standard elements also, but the learning curve is steep and supporting documentation is only marginally helpful. However, personal experience indicates that it is possible to go from zero knowledge of *ns-2* to a working level in approximately two weeks. This is largely possible due to the fact that *ns-2* is open source–access to the source code is invaluable. This type of aid is not available with the proprietary OPNET package. There is also an on-line tutorial (46) that provides some useful information for *ns-2* beginners. Finally, network analysis is not automated in any way. *ns-2* is able to generate large amounts of simulation data but it is up to the developer to process the raw data for network performance data (such as network loading, queue length, bandwidth utilization, etc.).

C.3 Conclusion

When faced with numerous or large network modeling problems, *ns-2* provides a more efficient network modeling system. For the network modeling problem presented in this paper the run-time improvement factor of *ns-2* over OPNET ranges from approximately 3 (for 20 nodes) to more than 16 (for 100 nodes).

However, run-time performance is not the only issue to consider when determining what simulation environment to use. OPNET provides a robust and rich graphical development environment. Both network modeling systems have strengths and weakness. For simulation of large scale wireless networks, *ns-2* provides a more scalable environment. For faster, more intuitive system development and simulation, OPNET is better. The *ns-2* simulator could be improved significantly if a graphical font-end were developed to aid in network instantiation–however, this would not be a simple undertaking.

Much could be done to expand the performance analysis presented in this paper. While extensive testing was accomplished, the number of scenarios was limited. Also, the effect of network loading–as a result of changing the inter-arrival time of RESP messages or the number of pending RQST messages–on simulator performance was not explored. Additionally, there are parameters too numerous to mention that might affect simulator performance–including such things as the propagation and MAC models and their configuration parameters.

Appendix D. Summary of Swarm Algorithm Equations

The concepts and equations of the swarming algorithm of Chapter IV are summarized here for quick reference. For more detailed explanations, please see the appropriate sections in Chapter IV.

D.1 Swarm Algorithm

```
Loop $\forall p_i \in P, i = 1, .., N$
   Process boundaries
   Loop $\forall p_j \in P_i, j = 1, ..., N_i$
      Process neighbor $p_j$
      Calculate new direction
   end Loop
   Move in new direction
end Loop
```

Figure D.1 General Swarm Algorithm

Table D.1 Swarm Algorithm Variables

Variable	Description
P	The set of mobile particles
N	The population size (mobile particles), $\|P\|$
p_i	The i^{th} particle in P
P_i	The set of particles in p_i's *neighborhood* (includes way-points)
N_i	The number of particles in p_i's neighborhood, $\|P_i\|$
p_j	The j^{th} particle in P_i

D.2 Distance Dividing Points

Ideal separation distance is normalized to unity.

$$d = 1 \tag{D.1}$$

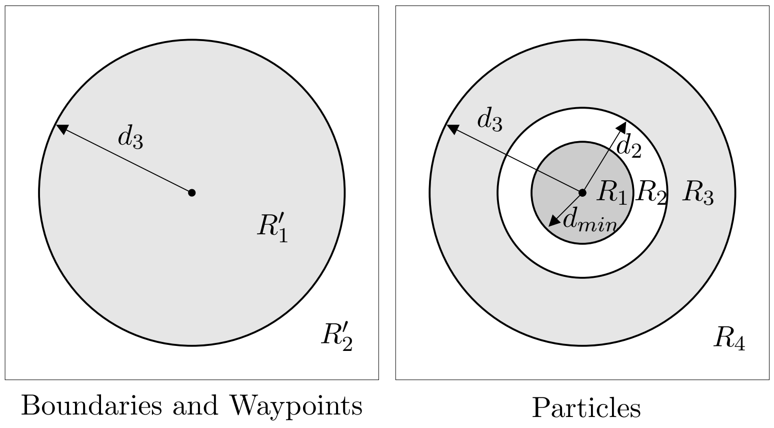


Figure D.2 Neighborhood models (see Figure 4.4)

The use of a *comfort zone* provides for a relaxation of the separation distance requirements. The requirements of *close* and *far* do not hold within the comfort zone region (see Table 4.4).

$$d_2 = 1 + c_{zone} \tag{D.2}$$

Particle are never so far way from each other that their influence on each other is zero. However, in the region *very far* (see Table 4.4), the inter-particle influence approaches zero.

$$d_3 = C_{max} d_2 \tag{D.3}$$

D.3 Particle Blocking Model

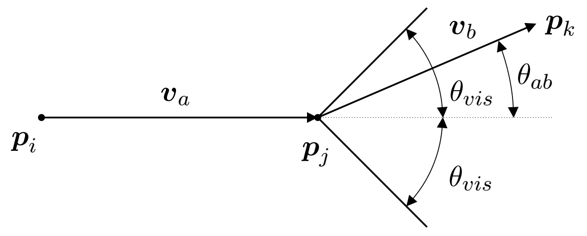


Figure D.3 Visibility Model (see Figure 4.2)

$$vis = \begin{cases} true & : \quad \theta_{ab} > \theta_{vis} \\ false & : \quad \theta_{ab} \leq \theta_{vis} \end{cases} \tag{D.4}$$

The value for θ_{vis} was chosen to be $\pi/3$.

$$\begin{aligned} \theta_{ab} &= \cos^{-1}\left(\frac{\boldsymbol{v}_a \cdot \boldsymbol{v}_b}{\|\boldsymbol{v}_a\| \cdot \|\boldsymbol{v}_b\|}\right) && \text{(D.5)} \\ \boldsymbol{v}_a &= \boldsymbol{p}_j - \boldsymbol{p}_i && \text{(D.6)} \\ \boldsymbol{v}_b &= \boldsymbol{p}_k - \boldsymbol{p}_j && \text{(D.7)} \end{aligned}$$

D.4 General Update Vector

$$\boldsymbol{v}_{update} = \sum_{N_i} \left[w_{periph} w_d \left(w_{attract} \boldsymbol{v}_{attract} + C_{align} \boldsymbol{v}_{align}\right)\right] \tag{D.8}$$

$$\begin{aligned} \boldsymbol{v}_{attract} &= \boldsymbol{p}_j - \boldsymbol{p}_i && \text{(D.9)} \\ \boldsymbol{v}_{align} &= \texttt{direction}(p_j) && \text{(D.10)} \end{aligned}$$

$$d = \|\boldsymbol{v}_{attract}\| \tag{D.11}$$

D.5 Boundaries

$$\boldsymbol{v}_{attract} = \boldsymbol{p}_b - \boldsymbol{p}_i, b \in \{north, south, east, west\} \tag{D.12}$$

$$w_d = \begin{cases} 0 & : \ p_b \in R'_2 \quad (d < d_3) \\ \sqrt{d_3 - d} & : \ p_b \in R'_1 \quad (d \geq d_3) \end{cases} \tag{D.13}$$

$$w_{attract} = \begin{cases} 0 & : \ p_b \in R'_2 \quad (d < d_3) \\ -C_{boundary} & : \ p_b \in R'_1 \quad (d \geq d_3) \end{cases} \tag{D.14}$$

D.6 Waypoints

$$\boldsymbol{v}_{result} = \boldsymbol{v}_{align}(\boldsymbol{v}_{attract} \cdot \boldsymbol{v}_{align}) \tag{D.15}$$

$$w_d = \begin{cases} \left(\frac{d}{d_3}\right)^2 \left(-\boldsymbol{v}_{align} \cdot \frac{\boldsymbol{v}_{result}}{d}\right) & : \ \boldsymbol{p}_{wp} \in R'_1 \quad (d < d_3) \\ \left(e^{\frac{-(d-d_3)}{d_3}}\right) \left(-\boldsymbol{v}_{align} \cdot \frac{\boldsymbol{v}_{result}}{d}\right) & : \ \boldsymbol{p}_{wp} \in R'_2 \quad (d \geq d_3) \end{cases} \tag{D.16}$$

$$w_{attract} = \begin{cases} -C_{wp} & : \ \boldsymbol{p}_{wp} \in R'_1 \quad (d < d_3) \\ C_{wp} & : \ \boldsymbol{p}_{wp} \in R'_2 \quad (d \geq d_3) \end{cases} \tag{D.17}$$

D.7 Peripheral Vision

$$w_{periph}(\theta) = C_{periph} \cos^n \left(\frac{\theta}{2}\right) \tag{D.18}$$

$$w_{periph}(\theta) = C_{periph} \left[\frac{1}{2}\left(1 + \cos\theta\right)\right]^{\frac{n}{2}} \tag{D.19}$$

A plot of the peripheral weighting is given in Figure D.4. This plot incorporates the distance weight w_d of Equation 4.22 (also shown in Equation D.20).

D.8 Particle to Particle

$$w_d = \begin{cases} \sqrt{1-d} & : \ \boldsymbol{p}_j \in R_1 \quad (d < d_{min}) \\ 0 & : \ \boldsymbol{p}_j \in R_2 \quad (d_{min} \leq d < d_2) \\ \left(\frac{d-d_2}{d_3-d_2}\right)^2 & : \ \boldsymbol{p}_j \in R_3 \quad (d_2 \leq d < d_3) \\ e^{\frac{-(d-d_3)}{d_3}} & : \ \boldsymbol{p}_j \in R_4 \quad (d \geq d_3) \end{cases} \tag{D.20}$$

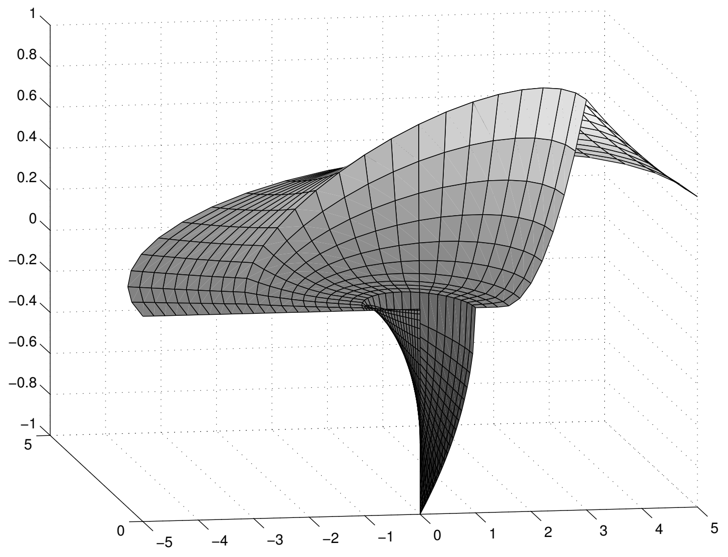


Figure D.4 Particle Attraction Weight

$$w_{attract} = \begin{cases} -C_{repulse} & : \ \boldsymbol{p}_j \in R_1 \quad (d < d_{min}) \\ 0 & : \ \boldsymbol{p}_j \in R_2 \quad (d_{min} \leq d < d_2) \\ C_{attract} & : \ \boldsymbol{p}_j \in R_3 \quad (d_2 \leq d < d_3) \\ C_{attract} & : \ \boldsymbol{p}_j \in R_4 \quad (d \geq d_3) \end{cases} \tag{D.21}$$

D.9 Movement

$$\theta'_{update} = \texttt{sgn}(\theta_{update} + \theta_p)\texttt{min}(\theta_{max},\ |\theta_{update} + \theta_p|) \tag{D.22}$$

$$\theta'_i = \|\theta_i + \theta'_{update}\|_{360} \tag{D.23}$$

The new update vector $\boldsymbol{v}'_{update}$ has the same magnitude as the update vector $\boldsymbol{v}_{update}$ above but its directions has been adjusted to θ'_i.

$$\boldsymbol{p}'_i = \boldsymbol{p}_i + \alpha_s \left(\boldsymbol{v}_{dir} + \frac{\boldsymbol{v}'_{update}}{N_i} \right) \tag{D.24}$$

Appendix E. Swarm Parameter Experimental Techniques

The values of the constants used in the swarm model are determined experimentally. This process is subjective and labor-intensive. The constants determined in this way and their values are presented in Equations E.1 through E.5.

$$C_{boundary} = 30 \pm 5 \tag{E.1}$$

$$C_{periph} = 1 \tag{E.2}$$

$$C_{wp} = 10 \pm 5 \tag{E.3}$$

$$\theta_{max} = 4^o \pm 2^o \tag{E.4}$$

$$R_{max} = 4 \tag{E.5}$$

Experimentation to determine the weighting factors involved running the GUI-based simulator numerous times starting with an initial estimate of the correct parameter values. The parameter values are perturbed slightly in order to obtain a desired behavior such as *coherent* or *incoherent*. Once a desired behavior is obtained, each parameter is varied individually while the remaining parameters are kept fixed. By visually inspecting the results of the GUI-based simulator, a range of parameter values that continue to produce the desired behavior is obtained. Typically, these experiments are repeated 20 to 30 times.

The C_{periph} and R_{max} parameters are exceptions. The value for C_{periph} is chosen to be unity since other weighting factors are involved. The R_{max} parameter is chosen to be 4 in order to limit the neighborhood effects of distant particles.

The parameters discussed thus far are constant for any swarm behavior. Table E.1 (repeated from Table 6.3). As stated, the determination of the values and their ranges is

Table E.1 Behavior Parameter Sets

Parameter	Behavior	
	Coherent	*Incoherent*
C_{align}	8 ± 2	0
$C_{repulse}$	8 ± 2	24 ± 6
$C_{attract}$	6 ± 2	12 ± 3
c_{zone}	0.1 ± 0.02	0.9 ± 0.2
α_s	0.008 ± 0.002	0.012 ± 0.003

subjective. A heuristic is used based on a percentage of 25% of the noted parameter value that obtained the "best" behavior. It was observed that this method is conservative in determining the range. However, since a sharp demarkation between the two behaviors is desired, conservative ranges are acceptable.

The exception to this rule is the C_{align} value for *swarm* behavior. The parameters are limited to non-negative values and small variations–on the order of 10^{-3}–of C_{align} above zero resulted in significant changes in behavior.

This same method was used to determine the packet sizes (see Section 7.2) used to test the communications protocols. Larger values cause traffic bottlenecks that result in dropped packets because of excessive collisions. This is undesirable since the efficiency of the protocols are being investigated.

The implementation of the swarm simulator is described in Section 4.2. Instructions for using the simulator are given here. The simulator is implemented as a graphics-based visualization tool and a command-line version. The command-line version facilitates batch mode processing.

F.1 Swarm Simulator Setup

The graphics-based simulator is controlled using menus and a toolbar. The menu functionality is described in Section F.1.1 and use of the toolbar is described in Section F.1.2.

F.1.1 Menus. The menu items are shown in Table F.1. They provide some of the traditional support such as file access methods as well as control of the swarm parameter settings and simulation execution. The main menu items are `File`, `Edit`, `Swarm`, `View`, and `Help`. The `Help` menu item is not implemented. The submenus for each of the main menu items is described in the following sections. The *MRU List* under the `File` menu is a list of

Table F.1 Menu Commands

File	Edit	Swarm	View
New	Set Params...	Initialize	Toolbar
Open...	Set Region Size...	Start Thread	Status Bar
Save	Clear Region	Stop Thread	Swarm Info...
Save As	Set play speed...	Play	Refresh
Import Swarm...		Pause	Show Direction
Save History...		Step forward	
Set Param File...		Step backward	
Print...			
Print Preview			
Print Setup			
MRU List			
Exit			

up to four of the most recently used (MRU) formation files. When an item with an ellipsis (...) is selected, a dialog box is opened.

File. The `File` menu items are responsible for general file management. The first four items refer to a file formation file (extension `fmn`). The particle formation for a simulation can be saved to a data file and subsequently read in for the initial location. It is important to note that the formation data that gets saved is the location and direction information when the simulation is stopped.

The `Import Swarm...` item allows a text file containing swarm position and direction information to be loaded. This is useful when a particular formation configuration is desired. Selecting this item opens a File Open dialog box that can then be used to navigate to and select the file to be read in. The format of the file is given in Figure F.1. The lines

```
Size:  10
Waypoints:  0
RDist:  80.000000
Comfort Zone:  0.100000
Move dir:  0.000000
Move steps:  100
Velocity:  5.000000
Entropy:15.000000
Maxturn:  5.00000
Iterations:  3000
Seed:  1952
0:  n 321.114562 -471.114562 1.000000 0.000000
1:  n 321.114562 -609.678626 1.000000 0.000000
2:  n 361.114562 -540.396594 1.000000 0.000000
3:  n 361.114562 -678.960659 1.000000 0.000000
4:  n 401.114562 -471.114562 1.000000 0.000000
5:  n 401.114562 -609.678626 1.000000 0.000000
6:  n 441.114562 -540.396594 1.000000 0.000000
7:  n 441.114562 -678.960659 1.000000 0.000000
8:  n 481.114562 -471.114562 1.000000 0.000000
9:  n 481.114562 -609.678626 1.000000 0.000000
```

Figure F.1 Sample Swarm Text File

below `Seed` provide the data for placing the particles. In this case there are 10 particles. The first column is an index, the second column is an `n` for a regular particle or a `w` for a waypoint. The third and fourth columns are the x and y coordinates respectively. The fourth and fifth columns are the x and y components of the direction vector. The magnitude of the direction vector is not required to be unity.

The `Save History...` menu item allows for the swarm simulation track data that is stored in memory to be written to a binary file (with extension `swh`). The format of this file is given in Figure F.2. The data types are shown also. The *scale* value specifies what

```
< float   scale >
< int   L > {
  < int   N > {
    < double   x, y, d >
  }
}
```

Figure F.2   Binary File `swh` Format

the *ideal* separation distance for the particle swarm. The L value specifies the number of time steps in the file. For every time step, there is data for the number of particles, N, and location and direction data for every particle. The file size can be calculated according to Equation F.1.

$$S = 8 + L(4 + 24N) \tag{F.1}$$

The `Set Param File...` menu item allows for dynamic parameters to be specified. This data is contained in a text file with an extension of `dyn`. This item is useful for generating multiple types of behaviors for a single swarm simulation. The format of this file is given in Figure F.3. The parameters in the `dyn` file are described in Chapter IV.

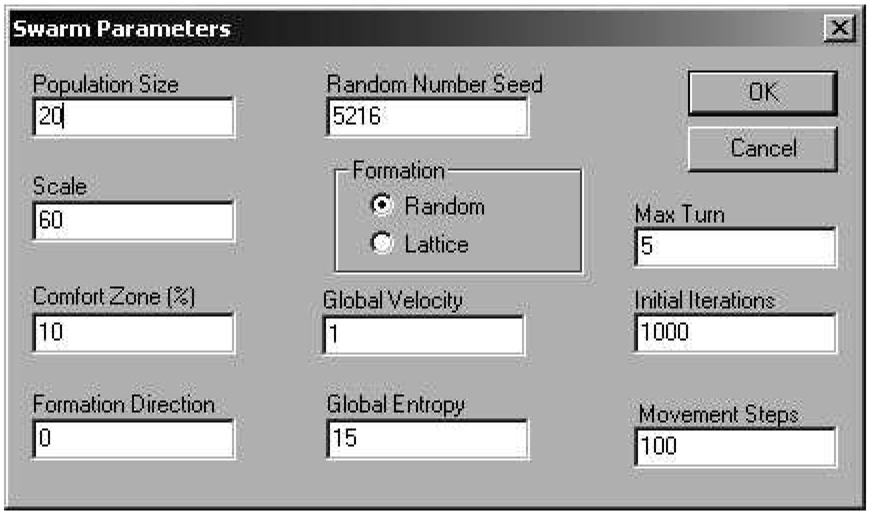
```
; Lines that begin with a semi-colon are ignored
;
; Parameters: (typical values)
;   A     - Boundary repulse weight        30.0
;   B     - Periph weight                  10.0
;   C     - Waypoint weight                20.0
;   D     - Repulse weight                 30.0
;   E     - Alignment weight                0.5
;   F     - Linear bias (slope)             0.5
;   G     - Attraction weight               1.0
;   N_h   - Neighborhood size (int)         7
;   v_fac - velocity factor                 0.2
;   t_max - max turn amount (degrees)       5.0
;   czone - comfort zone [0, 1]             0.1
;   p_max - max turn perturb                2.0
;
;    t    A    B    C    D   E   F    G  N_h   v_fac t_max czone p_max
;------ ---- ---- ---- ---- --- --- ---- ---   ----- ----- ----- -----
;
   0.0 10.0  1.0 20.0  8.0 6.0 0.0  6.0   7   0.008  4.0   0.1   4.0
 500.0 10.0  1.0 20.0 24.0 0.0 0.0 12.0   7   0.012  4.0   0.9   4.0
; End of file, file must end with at least one comment line
```

Figure F.3 Sample dyn Text File

The remaining menu items–Print..., Print Preview, and Print Setup–provide the mechanism for printing a hardcopy of the swarm simulation

Edit. The Edit menu subitems allow various simulation parameters and settings to be specified. The Set Params... item opens a dialog box (shown in Figure F.4) where some of the swarm parameters can be set. These settings are overridden by the dynamic parameters when specified.

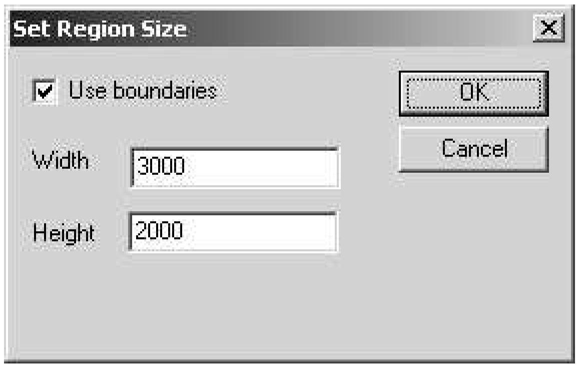
The Set Region Size... item allows the size of the region in which the swarm can move. The dialog box where these values are set is shown in Figure F.5. The mapping mode used converts logical units to 0.1 millimeters. This conversion is only significant for printed output. Since the simulation is normalized to unit distance separation, the resulting track data can be scaled to any desired system. Alternatively, the boundaries can be disabled by removing the check in the Use boundaries checkbox.



Swarm Parameters

Population Size	20
Random Number Seed	5216
Scale	60
Formation	[✓ Random] Lattice
Max Turn	5
Comfort Zone (%)	10
Global Velocity	1
Initial Iterations	1000
Formation Direction	0
Global Entropy	15
Movement Steps	100

Figure F.4 Set Parameters Dialog Box



Set Region Size

Use boundaries	✓
Width	3000
Height	2000

Figure F.5 Set Region Size Dialog Box

The `Clear Region` command clears the screen of any track plotting and repaints the swarm formation with edges as described above. This command can be used even during the simulation to clear away the clutter of track plots.

The `Set Play Speed...` command opens a dialog box that lets the user specify a time duration to use between updates when playing back the swarm history.

Swarm. The Swarm menu items are used to manage the threads that provide the simulation and history playback.

The `Initialize` command constructs the swarm data structure using the parameters specified earlier (with either the `Set Params...` command or the dynamic parameter file).

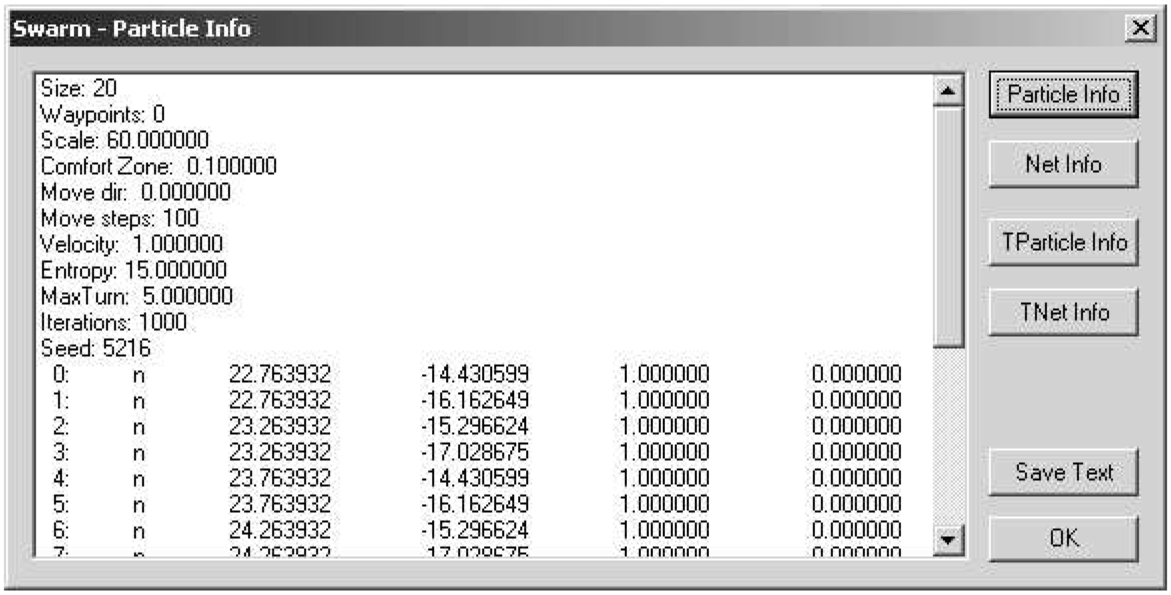
The `Start Thread` and `Stop Thread` command respectively start and stop the *simulation* thread. The simulation thread is main computational element of the swarm simulator executable. It encompasses all the computations necessary to implement the algorithm as described in Chapter IV.

The remaining commands–`Play`, `Pause`, `Step forward`, and `Step backward`–manage the *playback* thread. This thread allows the user to review the stored swarm track data. As the track data is replayed, the connectivity (as described above) shown. The `Play` command starts a continuous playback. The `Pause` command interrupts the playback. By using the `Play` command again, the animation is started from where it stopped. The two command `Step forward` and `Step backward` enable the user to step through the track data one time unit at a time going forward or backward in time.

View. The View menu items manage the simulator environment. The first two commands–`Toolbar` and `Status Bar`–are toggles that turn the toolbar and status bar on or off. The toolbar provides shortcuts to several of the menu items described above. The status bar provides information on the menu items and toolbar buttons when the mouse if located over them. The functionality of the toolbar is described in detail in Section F.1.2.

The `Swarm Info...` command opens up a dialog box, shown in Figure F.6, that displays latest information on the swarm. Two different types of data can be displayed (and saved to a text file). First, data about the swarm location and direction can be displayed (`Particle Info` and `TParticle Info` buttons). The data shown in Figure F.6 is a result of selecting the `Particle Info` button. This data has the exact same format as that described for the `Import Swarm...` command. The `TParticle Info` buttons displays the data for the swarm at time equal zero (after initialization, before beginning simulation) and does not include the header information. The second format provides details concerning the *network*. The difference between `Net Info` and `TNet Info` is the same as for the particle information. The format of the data for the net information is given below.

```
index type x y dist neighbor out-degree
```



```
Size: 20
Waypoints: 0
Scale: 60.000000
Comfort Zone:  0.100000
Move dir:  0.000000
Move steps: 100
Velocity:  1.000000
Entropy: 15.000000
MaxTurn:  5.000000
Iterations: 1000
Seed: 5216
 0:     n      22.763932      -14.430599      1.000000      0.000000
 1:     n      22.763932      -16.162649      1.000000      0.000000
 2:     n      23.263932      -15.296624      1.000000      0.000000
 3:     n      23.263932      -17.028675      1.000000      0.000000
 4:     n      23.763932      -14.430599      1.000000      0.000000
 5:     n      23.763932      -16.162649      1.000000      0.000000
 6:     n      24.263932      -15.296624      1.000000      0.000000
```

Figure F.6 Swarm Information Dialog Box

The `index`, `type`, `x`, and `y` fields are the same as described above for the format of the swarm import text file. The `dist` field is the distance to the nearest neighbor whose index is given by the `neighbor` field. The `out-degree` field specifies how many other particles are within $1 + c_{zone}$.

F.1.2 Toolbar. As stated in the previous section, the toolbar provides shortcuts to several menu commands. The details of these shortcuts are described here. The toolbar buttons are shown in Figure F.7. A summary of their functionality is described in Table F.2. The `Cut`, `Copy`, `Paste`, and `Help` functions are not implemented.

F.2 Command line simulator

A command line version of the simulator was developed to facilitate batch file processing. The command line executable allows for the following switches: The program requires a text file with swarm configuration data to be present in a file named `params.txt`. The format of this file is shown in Figure F.8. The file must have the exact format shown. The `Region` data is the x and y dimensions of the swarm region as described above for the `Set Region Size...` menu command. The `Popsize` parameter specifies the number of

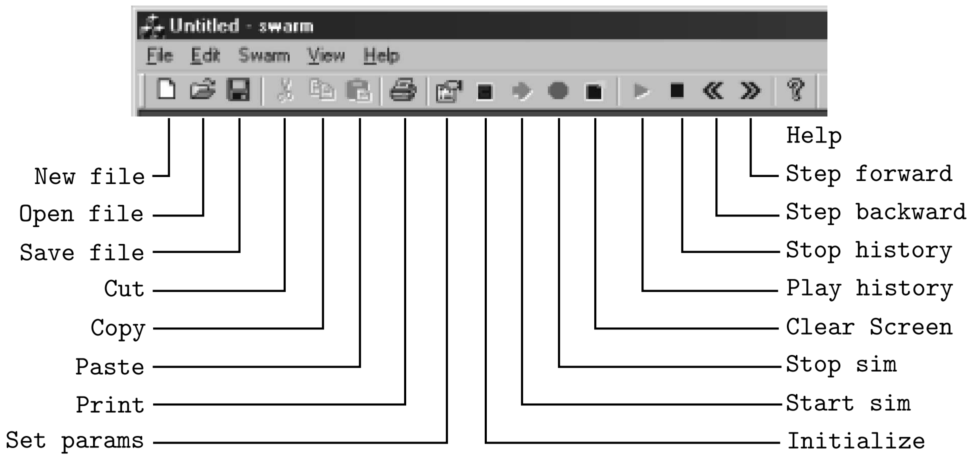


Figure F.7 Simulator Toolbar

```
Region 3000 2000
Popsize 15
CZone 10.0
Dir 0.0
Seed 5216
Type 1
Turn 5.0
```

Figure F.8 Command Line Parameter File

swarm members. The `CZone` parameter is a percent value that specifies the value for c_{zone} in Equation 4.24.

The `Dir` parameter specifies the initial direction of all the particles. This value is in degrees and is measured counterclockwise from the positive x-axis.

The `Seed` parameters specifies the random number generator seed value. The `Type` parameter is zero for an initially random formation and one for a initial lattice formation. The `Turn` parameter specifies the maximum turn angle (in degrees).

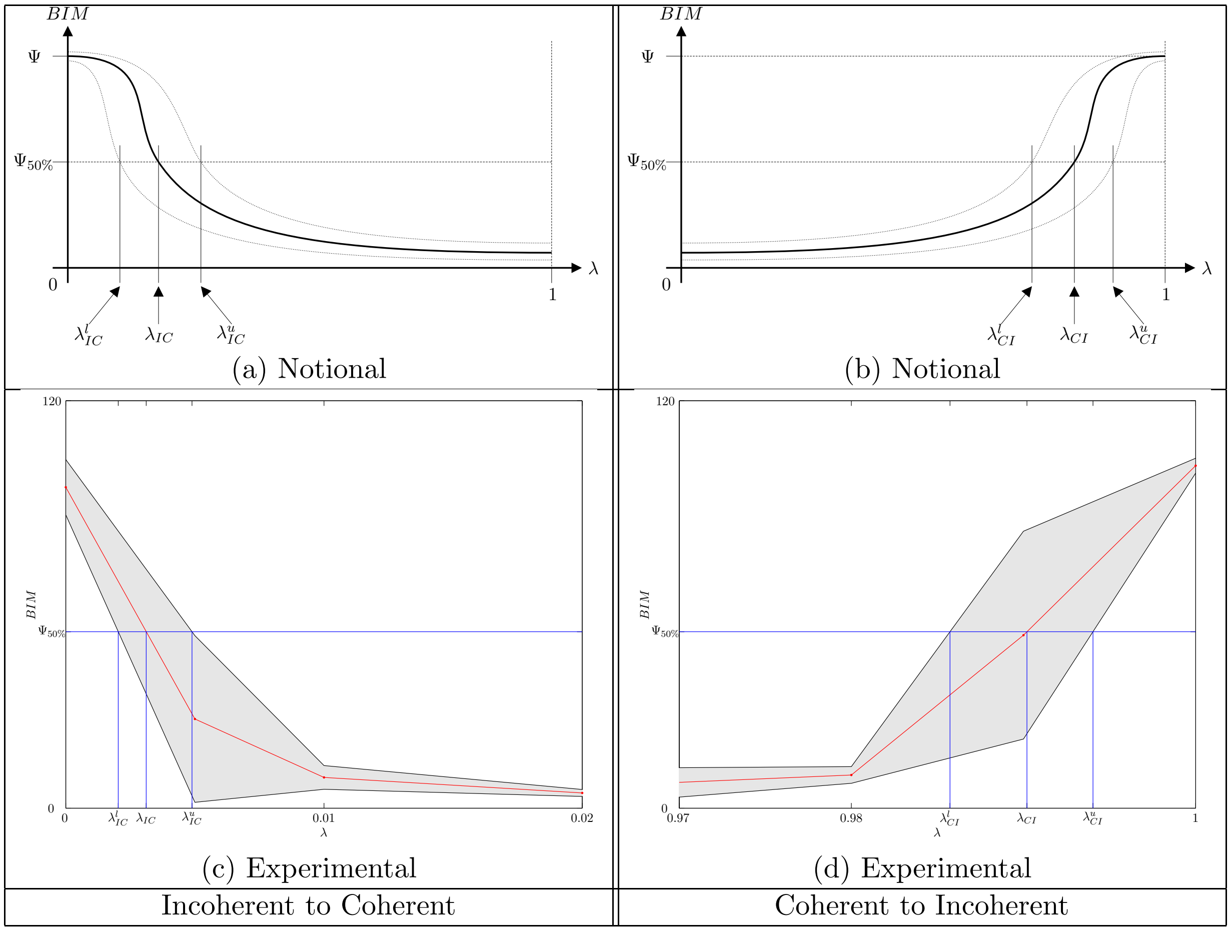
Table F.2 Toolbar Shortcut Summary

Command	Description
`New`	Prepares the simulator for a new simulation by clearing the screen and memory
`Open...`	Opens a previously saved *formation* to be used as the initial starting point for a new simulation
`Print`	Prints the currently display swarm simulation
`Set Params...`	Allows some of the parameters (number of particles, speed, turn radius, etc.) to be modified
`Initialize`	Initialized the swarm (with random particle placement) based on the parameters set above
`Start Thread`	Once the swarm is initialized, this will turn green. Clicking this button starts the simulation
`Stop Thread`	This stops the simulation
`Clear Region`	This clears the simulation
`Play`	As the simulation runs, track information is stored in memory. When you click the 'stop' button above, this 'play' button will be enabled. It allows the track information stored in memory to be played back
`Pause`	This stops the playback (but doesn't reset the 'playback pointer' to the beginning–it's essentially a 'pause' button)
`Step backward`	While 'paused', the playback pointer can be moved back in 'time'
`Step forward`	While 'paused', the playback pointer can be advanced in 'time'

Table F.3 Command Line Switches

Switch	Typical	Description
`/p`	`temp.dyn`	
`/h`	`temp.swh`	
`/m`	`temp.met`	
`/s`	191	
`/n`	20	
`/i`	4000	
`/b`	`no`	Use specified boundary (`yes`|`no`)

This appendix presents the experimental results of *noise margin* analysis as described in Section 6.2.3. The example threshold plots of Figure 6.19 are repeated here in Figure G.1(a) and (b) for convenience.



(a) Notional (b) Notional

(c) Experimental (d) Experimental

Incoherent to Coherent Coherent to Incoherent

Figure G.1 Input and Output Thresholds

The experimental plots are presented in Figure G.1(c) and (d). The shaded regions represent one standard deviation above and below the average BIM values for λ (see Section 6.2.3) over the specified ranges. The statistical data is obtained from 10 separate and independent runs for each value of λ. The experimental values for the incoherent behavior (IB) to coherent behavior (CB) and CB to IB thresholds λ_{IC} and λ_{CI} respectively are obtained by determining where the plots cross the $\Psi_{50\%}$ lines. The same is true for the upper and lower values for each threshold. The values for the *CB to IB* and *IB to CB* thresholds are given in Tables G.1 and G.2 respectively.

Table G.1 Parameter Threshold Values

Coherent to Incoherent		
λ^l_{CI}	λ_{CI}	λ^u_{CI}
0.0020332	0.0031178	0.0048882

Table G.2 Parameter Threshold Values

Incoherent to Coherent		
λ^l_{IC}	λ_{IC}	λ^u_{IC}
0.98573	0.99020	0.99404

It should be noted that, even though the data points of the experimental data in Figure G.1(c) and (d) are connected by continuous lines, it is a weak assumption that the use of linear interpolation is valid for obtaining the threshold values. The nature of swarm behavior is highly stochastic. As a result, these values are merely rough approximations. The computed experimental noise margins, as determined by Equations 6.16 and 6.17 are given below.

$$NM_C = 51.626$$
$$NM_I = 50.932$$

Appendix H. Connectivity Analysis

In addition to the Behavior Identification Metric (BIM) which measures the amount of dynamic behavior of the swarm, the communications metrics for topology changes and node connectivity provide a direct linkage between swarm behavior and network configuration. A detailed discussion of the dynamics of swarm networks is provided.

Two measures of network dynamics are used to characterize each scenario. The first metric measures the rate of change in the network while the second measures the connectivity between the data source and the sink as a percentage of time.

H.1 Network Connectivity

The numerical data for the two measures of network dynamics is provided in Table H.1. Plots for each swarm scenario is provided in Figures H.1 and H.2. The flock scenario, because of its lack of dynamics, is not included. The random number seed value is used for the name of each scenario. In the case of scenarios 5 through 10, the additional index indicates the subset data of the simulation run.

Table H.1 Swarm Scenario Dynamics

Scenario	Name	Topology Changes (s^{-1})	Connection Duration (%)
1	1934	3.49	72.88
2	5935	4.33	81.70
3	6449	3.66	62.56
4	8216	6.16	100.00
5	7184_1	4.80	29.97
6	7184_2	3.20	49.45
7	7184_3	4.10	70.33
8	9582_1	4.40	51.25
9	9582_2	3.50	55.24
10	9582_3	2.30	59.24

The numerical data of Table H.1 provides a fast method for determining network dynamics but fails to capture the instantaneous variations in the network over time. The plots of the network dynamics provide a means for a visual evaluation of this network quality. The dots represent points in time when the network connectivity between the source and sink nodes changes. The solid lines indicate whether there is a path between the source and sink nodes. The value is unity when the path exists and zero otherwise.

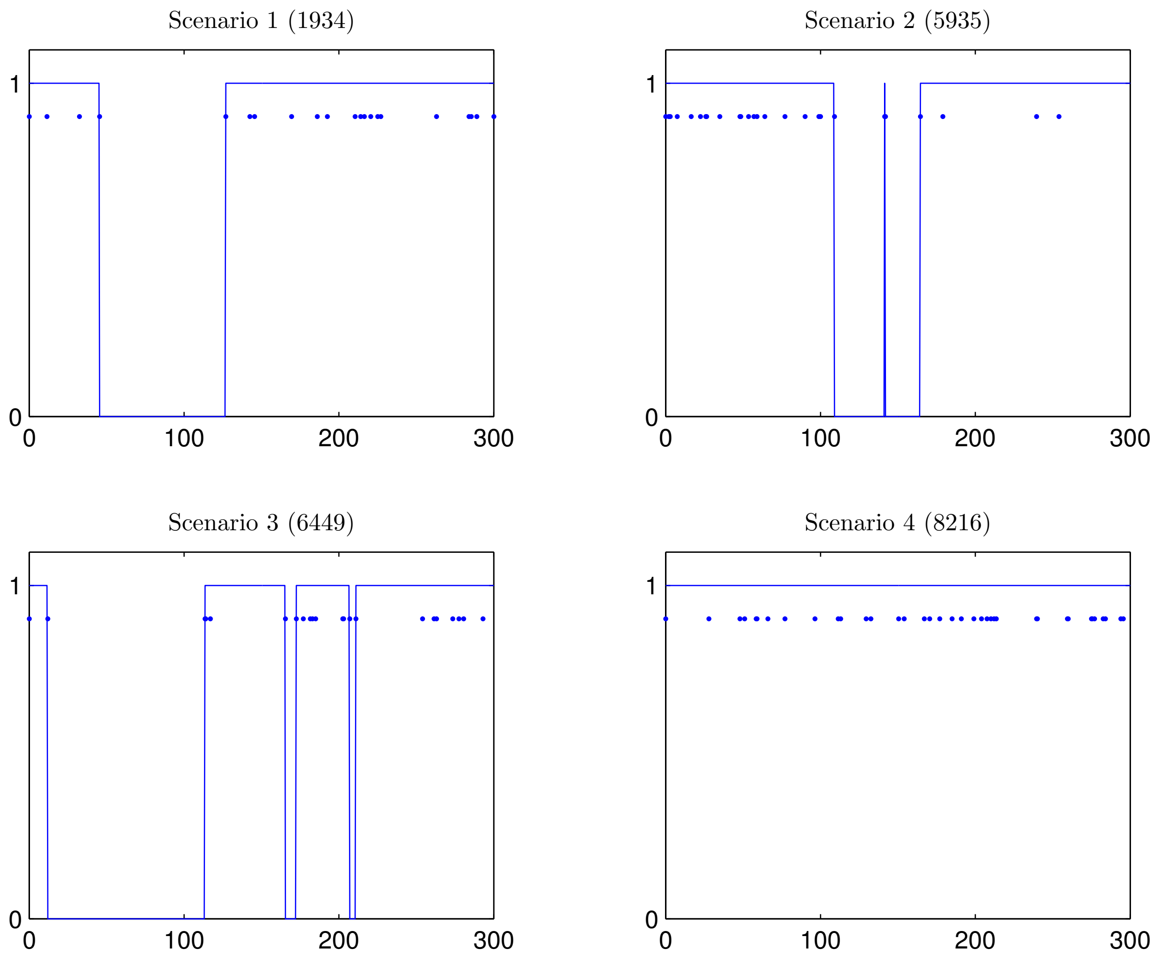


Figure H.1 Connectivity, Scenarios 1 through 4

As the figures indicate, there is a wide range in the change rate and path duration measures. These scenarios provide a wide range of network configurations for testing the protocols in a sensor swarm network.

Tables H.2 through H.4 contain the raw data for the results presented in Section 7.2. The scenario numbers in column one match the scenario numbers of Table H.1 with the exception of scenario zero which is the Flock scenario.

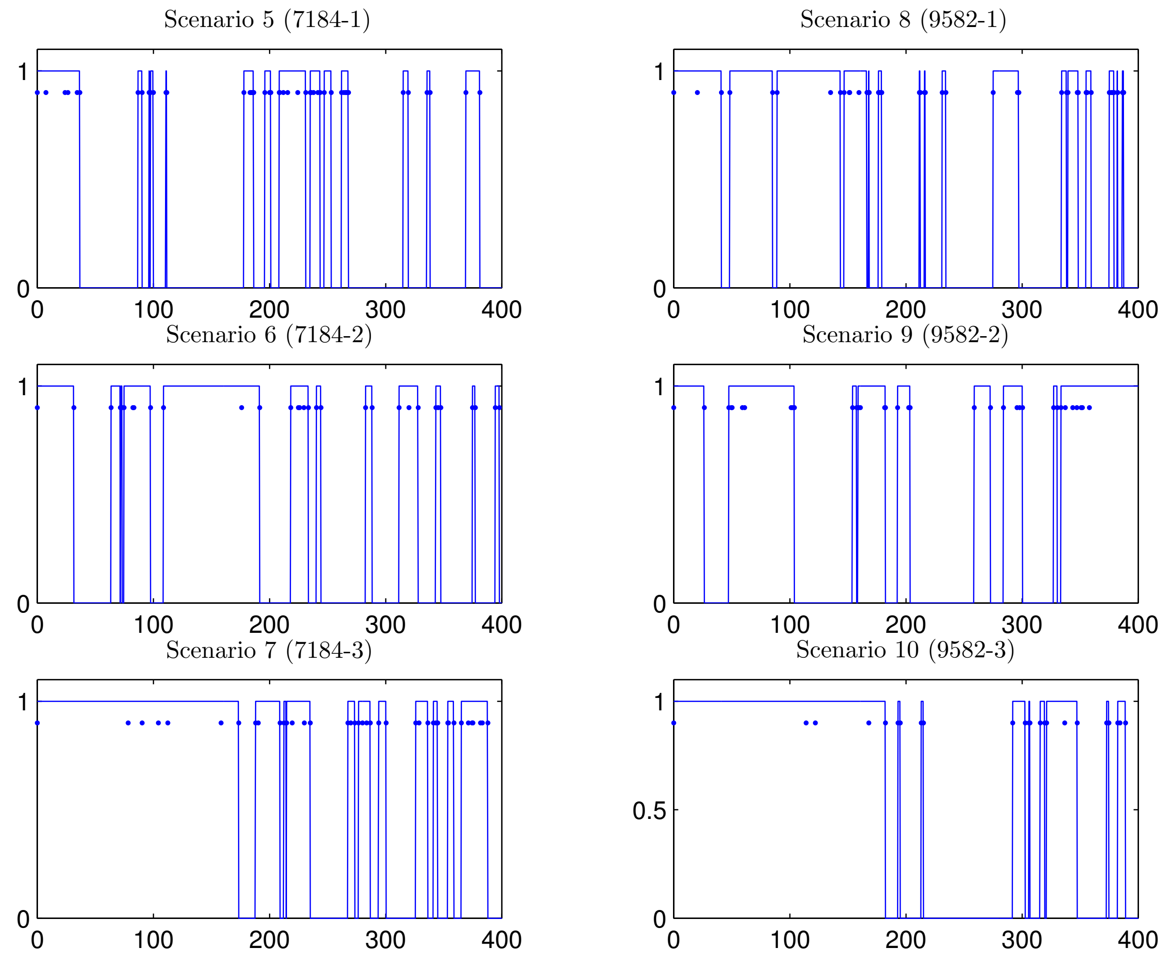


Figure H.2 Connectivity, Scenarios 5 through 10

Table H.2 Delivery Effectiveness, Raw Data

Scenario	sDiff	GRP	Flood
0	1	1	1
1	0.57780157	0.660242684	0.687009279
2	0.782547002	0.627527492	0.769776516
3	0.96744186	0.984883721	0.969767442
4	0.876074499	0.776146132	0.953796562
5	0.20775112	0.23068811	0.249143158
6	0.451308901	0.426701571	0.451832461
7	0.635411069	0.623858141	0.656636217
8	0.627987254	0.401221455	0.461763144
9	0.488323275	0.445027552	0.513513514
10	0.544790257	0.548037889	0.551014885

Table H.3 Control Overhead Efficiency, Raw Data

Scenario	sDiff	GRP	Flood
0	0.828063719	0.546593797	0.818256000
1	0.886457808	0.744982316	0.982531444
2	0.819264374	0.764576051	0.804708324
3	0.806060606	0.640184603	0.787962844
4	0.850046777	0.805889647	0.838250412
5	0.958771612	0.849911607	0.924144593
6	0.899001062	0.697854522	0.867262645
7	0.892471793	0.659010994	0.882403257
8	0.898277382	0.74579882	0.875092889
9	0.913235327	0.76575981	0.891838149
10	0.928688507	0.722015281	0.915659593

Table H.4 Data Throughput Efficiency, Raw Data

Scenario	sDiff	GRP	Flood
0	5.055131583	6.143504176	20.47576904
1	8.929575307	11.27818834	22.48721591
2	7.081981669	13.92827096	18.17445817
3	5.502847525	8.367768595	20.16724511
4	8.736780873	14.77849562	19.75118742
5	31.39543098	17.52304688	37.68337674
6	9.297509426	8.642218654	20.9777337
7	6.639675277	6.538300428	20.55270481
8	5.544480048	8.624546296	17.02596567
9	10.85613728	9.023558419	19.63227457
10	10.02875838	7.270399306	21.4134591

H.2 Behavior Identification Metric by Scenario

The network dynamics are related to swarm behavior characteristics as described in Section 7.2.3. This section provides supporting data.

The average BIM across the swarm members at each time step for each of the ten scenarios is plotted in Figure H.3. The horizontal axis in each plot is time and the vertical axis is the average BIM. Scenarios 1 through 4 indicate a slightly less varying BIM over time compared to scenarios 5 through 10.

The overall means of each BIM for each scenario (and the standard deviation) are presented in Table H.5. This data is plotted in Figure 7.8.

Table H.5 Overall Mean BIM by Scenario

Scenario	Mean	St. Dev.
1	92.7371	12.2235
2	96.1251	6.8339
3	102.0796	5.1721
4	86.8204	8.5004
5	87.5271	28.7361
6	75.1390	29.1058
7	63.7616	29.9976
8	80.1096	27.2911
9	75.4605	26.7417
10	74.1540	25.5645

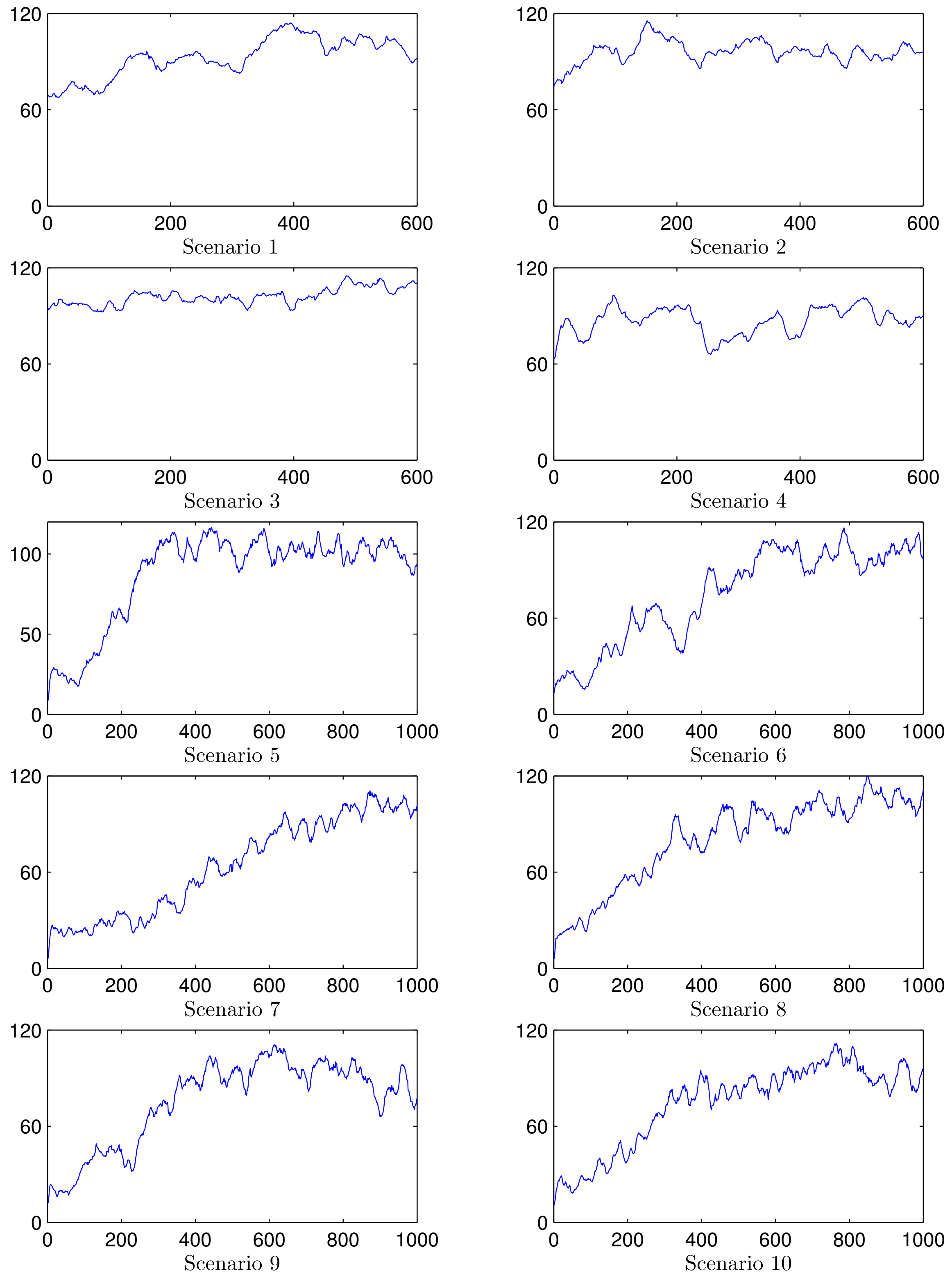


Figure H.3 Average BIM by Scenario

Appendix I. Software Tool Support

Numerous scripts are used to generate the swarm simulation and network communications data for testing and analysis. This appendix documents these scripts and their usage.

I.1 Swarm Simulator in Batch Mode

The `test.bat` batch file runs the swarm simulator for each random number in each of six random number files.

Batch mode test.bat Script

```
@echo off
if exist %1 (
  echo filename res%1.txt exists
  goto DONE
)
for /f %%i in (rseed%1.txt) do (
  echo SEED %%i >> res%1.txt
  swarm /p long.dyn /h temp.swh /m temp.met /s %%i /n 20 /i 8000 /b no > res%1.txt
  transient long.dyn temp.swh temp.met >> res%1.txt
)
:DONE
```

The batch file has one command line argument that specifies which test to run–1 through 6. Using the following command causes the complete set to run.

```
for %i in (1 2 3 4 5 6) do (test %i)
```

The `gawk` script extracts the duration data from each of the `test.bat` output files. The output data tables have 10 columns with format as follows:

```
SEED UP1 DN1 UP2 DN2 UP3 DN3 UP4 DN4 UP5
```

Tables for the average transition times are generated in the same way except that `$3-$2` in lines 4 and 5 are replaced with `$4` and `tbl%i.txt` on line 18 is replaced with `avetbl%i.txt`.

```
Duration gawk Script
```

```
for %i in (1 2 3 4 5 6) do (
  gawk < res%i.txt "{ \
    if ($1==\"SEED\") print $0; \
    if ($1==\"UP\") print $1, $3-$2 \
    if ($1==\"DN\") print $1, $3-$2 \
  }" | gawk "BEGIN {c=0} { \
    if (c\%10==0) c=0; \
    print c, $0; \
    c++ \
  }" | gawk "{ \
   d[$1]=$3; \
    if ($1==9) { \
      printf \"%4d \", d[0]; \
      for (i=1; i<10; i++) \
        printf \"%7.2f\", d[i]; \
      printf \"\n\" \
    } \
  }" > tbl%i.txt
)
```

These tables can be loaded into MATLAB and analyzed with the Kruskal-Wallis m-file `kwallis.m`. The `kwallis` function returns an H value which may be considered to be approximated by chi-squared distribution with the degrees of freedom equal to one less than the number of groups(79).

The following batch file `bfsbat.bat` calls the `bfs` program to generate connection statistics for a swarm scenario (see Section 7.2 and Appendix H). The connectivity between nodes 4 and 14 is checked. It uses the `bfs` program to find the shortest path between the two nodes.

```
bfsbat.bat batch file
```

```
bfs %1 4 14 %2 | \
  gawk -F: '{if ($2!=prev) \
    print $1, 1, ":", $2; \
  else \
    print $1, 0, ":", $2; \
  prev=$2}' | \
  gawk -F: '{if (index($2, "no path")!=0) \
    print $1, 0; \
  else \
    print $1, 1}' > %3
```

There are three command line arguments to the batch file. The first is the name of the `swh` file, the second is the maximum edge length (corresponds to the maximum radio reception distance), and the third is the text filename in which the results are to be saved. A sample invocation of this batch file is given here:

```
bfsbat sw9582_1.swh 137.7 bfs9582_1.txt
```

This generates a file with three columns. The first column is simply a counter that is similar to "time". The value in the second column is a one when there is a change in the path between nodes 4 and 14. Otherwise it is zero. The value in the third column is a one when a path exists and a zero when the path does not exist.

The program `bfs` uses a breadth-first search algorithm (adapted from (109) and has the following command line invocation:

```
bfs filename.swh A B R
```

The `bfs` program reads the file `filename.swh` and prints the path from node `A` to node `B` for each time step if it exists. If the path does not exist, it prints `no path`. The parameter `R` specifies the maximum edge length.

I.2 Processing `swh` data in MATLAB

Several MATLAB scripts were developed for processing the raw swarm simulation data (hereinafter referred to as the `swh` data) for visualization or subsequent processing for use with the network simulator *ns-2*. The `swh` data is read using the `read_swh.m` script. The invocation of the function in MATLAB is shown below.

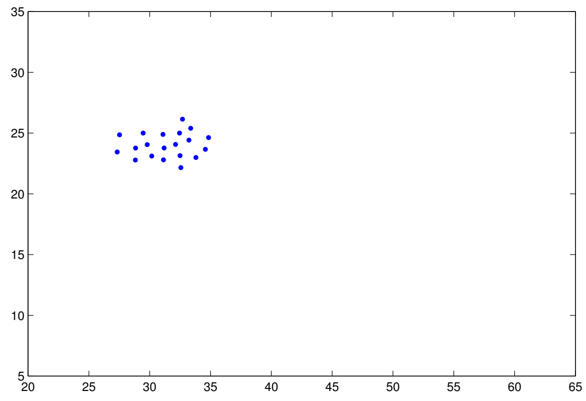
```
[x, y, d] = read_swh('filename.txt')
```

The x, y, and d data is stored as $N \times T$ matrices where N is the number of particles and T is the number of time steps. The x and y variables specify the location (in two dimensions) and the d variable specifies the direction (in degrees from the positive x-axis).

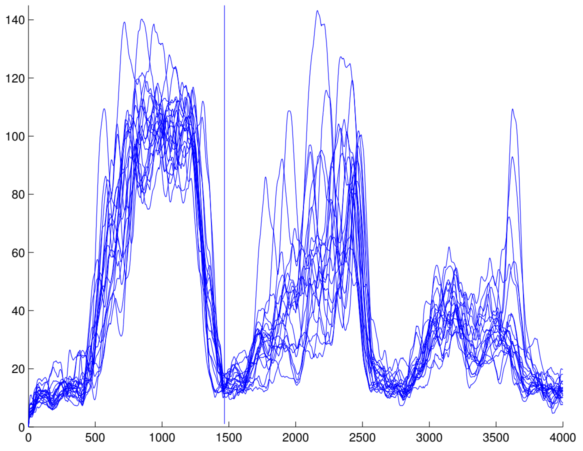
Writing a new `swh` data file from MATLAB is accomplished by using the `write_swh.m` script. The function is called as shown below. The x, y, and d variables are the same as those described for the `read_swh` function. The s variable specifies a scalar value that scales the input location data. For instance, if the original data was generated using unit separation distance but a scenario is desired where the ideal separation distance is 100 distance units, a value of 100 would be specified for s.

```
write_swh(x, y, d, s, 'filename.swh')
```

An animation of swarm movement is done with the `swarm_movie.m` script. This script opens two figure windows with one window showing the swarm movement as points in two dimensions and the other window showing the BIM metric (discussed in detail in Section 6.2.1). These two windows are shown in Figure I.1 for a 20 node swarm The figure



(a) Movement



(b) BIM Metric

Figure I.1 Swarm Animation

represents a snapshot in time for $t \approx 1490$. The time is indicated by the vertical line in Figure I.1(b). During the animation, this line sweeps from left to right as the swarm moves in the other window.

An animation of the connectivity is done with the `conn_movie.m` script. The script is called as follows. The x and y variables specify the location data (as described above for `read_swh`). The s variable specifies the ideal separation distance. This value is used to determine connectivity for the swarm as the animation progresses.

```
conn_movie(x,y,s)
```

A snapshot of the animation is shown in Figure I.2. Particles are indicated by the dots and links (when they exist) are indicated by lines connecting particles. This provides a visualization tool that can be used to gain insight into the nature of the network dynamics for a particular swarm scenario.

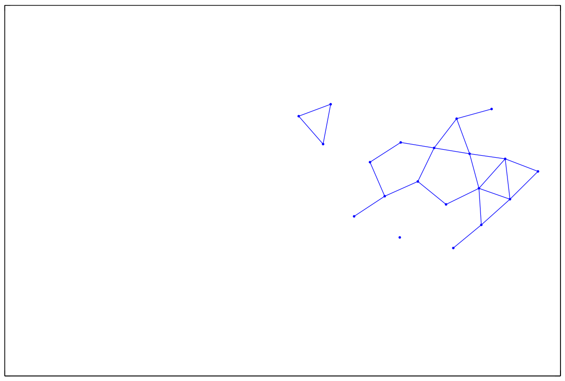


Figure I.2 Connectivity Animation

I.3 Movement scripts

This section describes the process to create an OTcl node movement script from the binary swarm simulation data. The raw swarm simulation data is normalized for unit separation distance. The `swh` data can be scaled to a desired separation distance using MATLAB. Additionally, any offset in the data can be removed in MATLAB as well. Once the `swh` data has been processed it is converted to the OTcl node movement file using the `gen-track` program. The command line for `gen-track` is as follows:

```
gen-track filename xoffset yoffset starttime timestep
```

The file name is specified without the `swh` extension. The values for `xoffset` and `yoffset` are used to translate the `swh` data in the x and y directions. Additionally, the start time for node movements must be set by specifying a value for `starttime`. The time delta between movement updates is set by the `timestep` value. This can be used to control the speed of the swarm members for the *ns-2* simulations.

I.4 Network traffic analysis

The scripts describe here are used to generate the performance data for comparison of the various network protocols. Additionally, a shell script is used to generate a message traffic pattern file from the sDIFF simulation (for a particular movement file) for use with the GRP and Flood simulations. This ensures that the network loading is the same for all three scenarios.

```
Generate message traffic pattern
```

```
gawk < diff020.tr '{if ($1=="s" && $3=="_4_" && $4=="RTR") \
  printf "$ns_ at %f \"$snk1 make-request\"\n", $2;        \
if ($1=="s" && $3=="_14_" && $4=="AGT" && $NF=="DATA")     \
  printf "$ns_ at %f \"$src1 make-response\"\n", $2}' > traff.tcl
```

The file `traff.tcl` must be specified in the `options.tcl` file for the GRP and Flood simulations.

The following scripts are used to determine the number of packets successfully delivered and the number of bytes sent.

```
intr-ete
```

```
#! /bin/sh
# sample usage
#   intr-ete diff020.tr temp.txt temp2.txt
gawk < %1 '{if ($1=="s" && $3=="_4_" && $4=="RTR" && $NF=="INTR") \
  print $1, $2, $3, $19;                                          \
if ($1=="r" && $3=="_14_" && $4=="AGT" && $NF=="INTR")            \
  print $1, $2, $3, $19}' |                                       \
gawk -F_ '{print $1, $2, $3}' |                                   \
gawk -F: '{print $1, $2} > $2
gawk < $2 '{if ($1=="s") t[$5]=$2;                                \
  if ($1=="r" && t[$5]>0) {                                       \
    print $5, $2, t[$5], $2-t[$5];                                \
    t[$5]=-1;                                                     \
  }}' > $3
wc -l $3
gawk < $1 '{if ($1=="s" && $4=="MAC" && $NF=="INTR")              \
  print $2, $3, $8, $19, $NF}' |                                  \
gawk 'BEGIN {sum=0}                                               \
  {sum+=$3}                                                       \
  END {print "bytes: ", sum}'
```

The above script is used to generate data for sDIFF INTR messages. Data for sDIFF DATA messages can be obtained by using the same script with the following changes: on lines 4, 6 and 16 replace `INTR` with `DATA` and on line 4 replace _4_ with `_14_`. A file with these changes is saved in `data-ete`. These scripts generate two files (see line 3 above). The first file has an entry for every packet sent by the source node and received by sink node. The data includes the type of entry (s-send, r-receive), the time the packet was sent/received, the node, and the packet identifier. The packet identifier consists of a two-tuple number made up of the sending node number and a serial identification number.

The second file has an entry for every successfully delivered packet. Each entry consists of the packet serial identification number, the times the packet was sent and received, and the delta. This file is used to compute end-to-end delay statistics.

The script used for the GRP and Flood simulations is slightly different. The script for RQST messages is given below.

```
rqst-ete
```

```
 1 #! /bin/sh
 2 # sample usage
 3 #   rqst-ete geo020.tr temp.txt temp2.txt
 4 gawk < $1 '{if ($1=="s" && $3=="_4_" && $4=="AGT") \
 5   print $1, $2, $3, $19; \
 6 if ($1=="r" && $3=="_14_" && $4=="RTR" && $NF=="RQST") \
 7   print $1, $2, $3, $19}' | \
 8 gawk -F_ '{print $1, $2, $3}' | \
 9 gawk -F: '{print $1, $2}' > $2
10 gawk < $2 '{if ($1=="s") t[$5]=$2; \
11   if ($1=="r" && t[$5]>0) { \
12     print $5, $2, t[$5], $2-t[$5]; \
13     t[$5]=-1; \
14   }}' > $3
15 wc -l $3
16 gawk < $1 '{if ($1=="s") && $4=="MAC" && $NF=="RQST") \
17   print $2, $3, $8, $19, $NF}' | \
18 gawk 'BEGIN {sum=0} { \
19   sum+=$3; \
20 } END {print "bytes: ", sum}'
```

The script for RESP messages is obtained by doing the following: on line 4 replace _4_ with _14_, replace line 6 with the following:

```
if ($1=="r" && $3=="_4_" && $4=="AGT").
```

and on line 16 replace `RQST` with `RESP`. A file with these changes is saved in `resp-ete`. The formats of the output files are the same as described above.

I.5 Discrete Integration

The time-based efficiency of the routing protocols can be computed using the `data-int` and `resp-int` shell scripts (shown below).

`data-int shell script`

```
#! /bin/sh
# sample usage (after data-ete diff020.tr temp.txt temp2.txt)
#  data-int diff020.tr temp2.txt data_int.txt
gawk < $2 '{print "pkt", $2, NR}' > pkts.txt
gawk < $1 '{if ($1=="s" && $4=="MAC" && $NF=="DATA") \
  print $2, $3, $8, $19, $NF}' | \
gawk 'BEGIN {sum=0} { \
  sum+=$3; print "data", $1, sum}' > data.txt
sort -n -k 2 pkts.txt data.txt | \
gawk 'BEGIN {p=0; d=0} { \
  if ($1=="data") \
    d=$3; \
  if ($1=="pkt") \
    p=$3; \
  if (p!=0) \
    print $2, d/(p*$2*20); \
  }' > $3
rm pkts.txt data.txt
```

`resp-int shell script`

```
#! /bin/sh
# sample usage (after resp-ete geo020.tr temp.txt temp2.txt)
#  resp-int geo020.tr temp2.txt resp_int.txt
gawk < $2 '{print "pkt", $2, NR}' > pkts.txt
gawk < $1 '{if ($1=="s" && $4=="MAC" && $NF=="RESP") \
  print $2, $3, $8, $19, $NF}' | \
gawk 'BEGIN {sum=0} { \
  sum+=$3; print "data", $1, sum}' > resp.txt
sort -n -k 2 pkts.txt resp.txt | \
gawk 'BEGIN {p=0; d=0} { \
  if ($1=="data") \
    d=$3; \
  if ($1=="pkt") \
    p=$3; \
  if (p!=0) \
    print $2, d/(p*$2*20); \
  }' > $3
rm pkts.txt resp.txt
```

These scripts generate text data files with time in the first column and efficiency (in bytes per node per second) in the second column.

Glossary

Active Message	A communications system for networked sensors
Ad Hoc Network	Communications networks that have dynamic, sometimes rapidly-changing, random, multihop topologies which are likely composed of relatively bandwidth-constrained wireless links.
AFRL	Air Force Research Laboratory
AODV	Ad Hoc On-demand Distance Vector routing protocol
ATR	Automatic Target Recognition
AUSS	Advanced Unmanned Search System–a Navy UUV, the AUSS is an untethered UUV that is able to autonomously perform basic mission tasks such as transiting to a given location, hovering, and executing pre-programmed sonar and optical search patterns
AUV	Autonomous Underwater Vehicle
BER	Bit Error Rate
BIM	Behavior Identification Metric, provides a distributed behavior identification mechanism
Boundary	A boundary is used to constrain a swarm to a particular region. It is typically rectangular
CB	Coherent Behavior, swarm behavior that resembles the behavior exhibited by a flock of birds
CFV	Cost Function Value
DARPA	Defense Advanced Research Projects Agency
DD	Directed Diffusion
DSDV	Destination-sequenced Distance-vector routing protocol
DSSN	Distributed Smart Sensor Network
Free Swimmer	An Navy UUV with neural network controlled sensors and autonomous mission planning capabilities
FTP	File Transfer Protocol
GAaRP	Geographical Addressing and Routing Protocol
Global Behavior	Global behavior of the swarm
GLS	Grid Location Service
GPS	Global Positioning Satellite
GRP	Geographical Routing Protocol, a stateless communications routing protocol system for swarm based sensor systems
IB	Incoherent Behavior, swarm behavior that resembles the behavior exhibited by swarming insects

IP	Internet Protocol
ISO	International Standards Organization
ISR	Intelligence, Surveillance, and Reconnaissance
JBI	The Joint Battlespace Infosphere (JBI) is a an information management system to support military operations by providing an information dissemination infrastructure.
JBI Client	An access point to the JBI that either provides information (publish) or uses information (subscribe).
JBI GLobal Grid	The Global Grid consists of heterogeneous network communications systems and provides connectivity for the JBI
JBI Info. Object	An Information Object is the fundamental element of the JBI. Every piece of information is encapsulated by information objects
JBI Owner	The JBI Owner is the commander or the commander's information staff
JBI Platform	A Platform is the access point to the JBI for Clients. Clients are hosts that publish information objects or subscribe to information objects
JBI Publish	Information objects are made available to the JBI through publish operations
JBI Server	A Server provides support services to the JBI. These services include security and management
JBI Subscribe	Information objects are routed to users that need the information by invoking a subscribe operation
JBI User	A User is an entity that publishes information to the JBI or subscribes to information.
KW	Kruskal-Wallis test used to determine the relationship among several groups of sample data
LOCAAS	Low Cost Autonomous Attack System
MAV	Micro Air Vehicle
MOUT	Military Operations in Urban Terrain–army infantry operations is locations where manmade construction and high population densities are the dominant features
MRU	Most Recently Used
MSSMP	Multipurpose Security and Surveillance Mission Platform
NEST	Network Embedded Software Technology, a DARPA program to solicit research that addresses the technical challenges for resource-constrained networks of embedded nodes.

ns-2	An object oriented, discrete-event simulator for network communications systems, developed and distributed for free by the Information Sciences Institute of the University of Southern California
Odyssey	A class of Navy UUVs used for automous surveying as well as surveillance and reconnaissance missions
OPNET	A commercial network communications simulation system, initially developed by the Massachusetts Institute of Technology
OSI	Open Systems Interconnect
PDF	Probability Density Function
SensorApp	Sensor Application Component, used to provide a standard interface to the network communciations protocol system for sensors
SwarmApp	Swarm Application Component, used to provide a standard interface to the network communications routing protocol system for applications such as fusion or ATR
PIC	Particle in Cell
Redirection	Global swarm behavior modification due to external environmental influences
RESP	A RESP message is used to propagate response messages through an ad hoc network. Response messages are used to satisfy RQST messages with appropriate sensor data
RQST	A RQST message is used to progagate a request message through an ad hoc sensor network. Request messages establish the need for sensor data by the originating node
SAR	Synthetic Aperturue Radar
sDiff	Simplified Directed Diffusion communications routing protocol system for mobile ad hoc networking
SLURP	Scalable Location Update-based Routing Protocol
SSW	Smart Sensor Web
TCP	Transfer Control Protocol
UAV	Uninhabited Aerial Vehicle
UUV	Unmanned Underwater Vehicles
Waypoint	Waypoints are used to guide the *global* movement of the swarm along a desired route
XML	Extensible Mark-up Language

Bibliography

1. ABCNews.com. *Predator UAV Airborne Surveillance*, 2002. http://abcnews.go.com/sections/us/DailyNews/military_predator.html.

2. AeroVironment. *AV's "Black Widow" Micro Air Vehicle Gets Notice*, 1999. http://www.aerovironment.com/news/news-archive/blackwid1.html.

3. AeroVironment, Inc. http://www.aerovironment.com, 2002.

4. AFRL/IFTA, "The Embedded Information Systems Engineering Branch," September 2002. http://www.if.afrl.af.mil/div/IFT/IFTA/IFTA_home.html.

5. AFRL/VACC, "Control Systems Development and Application branch," November 2002. http://www.va.afrl.af.mil/DIV/VAC/VACC/vacc_index.html.

6. Air Force Technology. *Joint Surveillance and Target Attack Radar System*.

7. Augustsson, Peter, et al. "Creation Of A Learning, Flying Robot By Means Of Evolution." *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, edited by W. B. Langdon, et al. 1279–1285. New York: Morgan Kaufmann Publishers, 9-13 July 2002.

8. Bäck, T., et al., editors. *Evolutionary Computation, Volumes 1 and 2*. Philadelphia, PA: Institute of Physics Publishing, 2000.

9. Baldwin, Rusty O., et al. "Packetized voice transmission using RT-MAC, a wireless real-time medium access control protocol," *ACM SIGMOBILE Mobile Computing and Communications Review*, *5*(3):11–25 (2001).

10. Barr, Larine. "Dull, Dirty, and Dangerous," *Military Aerospace Technology*, *2* (2003).

11. Bender, Bryan. "DoD Eyes Sensors to Give 'Urban Canyon Visibility'," *Jane's Defense Weekly* (February 2000).

12. Birch, M. C., et al. "Cricket-based robots," *IEEE Robotics & Automation Magazine*, *9*(4):20–30 (Dec 2002).

13. Bluetooth Special Interests Group. *Specification of the Bluetooth System*, December 1999. Version 1.0B.

14. Bonabeau, Eric, et al. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford: Oxford University Press, 2000.

15. Broch, Josh, et al. "A performance comparison of multi-hop wireless ad hoc network routing protocols." *Proceedings of the fourth annual ACM/IEEE international conference on Mobile computing and networking*. 85–97. ACM Press, 1998.

16. Browne, Keen, et al. "Behavior Combination and Swarm Programming." *Proceedings of the 2001 RoboCup*, edited by A. Burk, et al. 499–502. 2001.

17. Brownlee, K. A. *Statistical Theory and Methodology in Science and Engineering*. New York: John Wiley & Sons, 1965.

18. Caceres, R. and L. Iftode. "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments," *IEEE Journal on Selected Areas of Communications*, *13*(5) (June 1994).

19. Captain, Sean. "Future Gear: Tiny Chips, Everywhere," *PC World.Com* (October 2002).

20. Cassinis, R. "Landmines Detection Methods Using Swarms of Simple Robots." *Proceedings of the International Conference on Intelligent Autonomous Systems*, edited by E. Pagello. 2000.

21. Cassinis, R., et al. "Strategies for Navigation of Robot Swarms to be used in Landmines Detection." *Proceeding of the Third European Workshop on Advanced Mobile Robots*. 1999.

22. Casti, John L. *Reality Rules: Picturing the World in Mathematics*. New York, NY: Wiley Interscience, 1992.

23. Chandler, Phillip and Meir Pachter. "Heirarchical Control for Autonomous Teams." *Proceedings of the AIAA Guidance, Navigation, and Control Conference*. August 2001.

24. Chien, Chao C. *Professional Software Developement with Visual C++ 6.0 & MVC*. Hingham, MA: Charles River Media, 2001.

25. Clough, Bruce. *UAV Swarming? So What are Those Swarms, What are the Implications, and How do We Handle Them?*. Technical Report, Control Systems Development and Applications Branch (AFRL/VACC), 2003.

26. Coello Coello, Carlos A., et al. *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York, NY: Kluwer Academic Publishers, 2002.

27. Cord, Thomas J., "The SkyTote Unmanned Air Vehicle." AFRL Horizons, June 2002. http://www.afrlhorizons.com/Briefs/Jun02/VA0201.html.

28. Corr, Michael G. and C. Okino. *A Study of Distributed Smart Sensor Networks*. Technical Report, Thayer School of Engineering, Dartmouth College, March 2000.

29. Corr, Michael G. and C. M. Okino. "Networking Reconfigurable Smart Sensors." *Proceedings of Enabling Technologies for Law Enforcement and Security4232*. November 2000.

30. Corson, S. and J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations." The Internet Society, January 1999. RFC2501.

31. Crichton, Michael. *Prey*. New York: HarperCollins, 2002.

32. Crombie, Duncan. *The Examination and Exploration of Algorithms and Complex Behavior to Realistically Control Multiple Mobile Robots*. MS thesis, Australian National University, 1997.

33. Deckmyn, D. "Bluetooth," *Computerworld*, *34*:73–76 (June 2000).

34. Defense Advanced Research Projects Agency, http://www.darpa.mil/ipto/Solicitations. *Network of Embedded Software Technology*, 2002. BAA #02-12.

35. Defense Advanced Researh Projects Agency, "Micro Air Vehicle Sets Endurance Flight Record." http://www.darpa.mil/body/NewsItems/pdf/WASP.pdf, 2002.

36. Deitel, Harvey M. and Paul J. Deitel. *How to Program C++*. Englewood Cliffs, NJ: Prentice Hall, 2002.

37. Dudek, Gregory, et al. "A Taxonomy for Mulitrobot Systems." *Robot Teams: From Diversity to Polymorphism* edited by Tucker Balch and Lynne E. Parker, A K Peters Ltd, 2002.

38. Estrin, Deborah, et al. "Next Century Chanllenge: Scalable Coordination in Sensor Networks." *Proc. of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*. 1999.

39. Fall, Kevin and Kannan Varadhan, editors. *The ns Manual*. VINT Projecct, Univ. of California, Berkley, CA, LBL, USC/ISI, and Xerox PARC, April 2002.

40. Finn, Gregory G. *Routing and Addressing Problems in Large Metropolitan-scale Internetworks*. Technical Report ISI/RR-87-180, Information Systems Institute, Univ. of Southern California, March 1987.

41. Flenner, Robert. *Jini and JavaSpaces Application Development*. Upper Saddle River, New Jersey: Pearson Education, 2001.

42. Fletcher, Barbara. "New Roles for UUVs in Intelligence, Surveillance, and Reconnaissance." *Proc. of the 9th Pacific Congress on Marine Science and Technology*. June 2000.

43. Franklin, Stan. *Coordination without communication*. Technical Report, Inst. For Intelligent Systems, Univ. of Memphis, April 2001. www.msci.memphis.edu/∼franklin/coord.html.

44. Gazi, Veysel and Kevin M. Passino. "Stability of One-Dimensional Discrete-Time Asynchronous Swarm." *Proc. Of the IEEE Int'l Symp. On Intelligent Control/IEEE Conf. On Control Applications*. 2001.

45. Goldfarb, Charles F. and Paul Prescod. *The XML Handbook* (2d Edition). New Jersey: Prentice Hall, 2000.

46. Greis, Marc, "Tutorial for the UCB/LBNL/VINT Network Simulator "ns"." On-line. http://www.isi.edu/nsnam/ns/tutorial.

47. Han, J. and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

48. Han, J., et al. "Spatial Clustering Methods in Data Mining: A Survey." *Geographic Data Mining and Knowledge Discovery* edited by H. Miller and J. Han, Taylor and Francis, 2001.

49. Harlow, F.H. "The Particle-in-cell Computing Method in Fluid Dynamics," *Methods in Computational Physics*, *3*:319–343 (1964).

50. Headquarters, U.S. Army. *Field Manual 7-10, The Infantry Rifle Company*, December 1990. Appendix L, Urban Operations.

51. Hebert, Jeffrey, et al. "Cooperative control of UAVs." *Proceedings of the AIAA Guidance, Navigation, and Control Conference*. August 2001.

52. Henderson, Thomas C., et al. "Smart Sensor Snow." *Proceedings of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*. October 1998.

53. Herwitz, Stanley. *UAV Coffee Project*. Clark University, 2002. www.clarku.edu/faculty/herwitz.

54. Hill, J., et al. *Active Messages Communication for Tiny Networked Sensors*. Technical Report, Univ. of California, Berkeley, March 2002.

55. Hillman, R. G., et al. "Modeling the Joint Battlespace Infosphere." *Proceedings of the 6th Intl. Command and Control Research and Technology Symposium*. June 2001.

56. *Wireless LAN medium access control (MAC) and physical layer (PHY) specification*. IEEE Std. 802.11, 1997.

57. Infrared Data Association. *Serial Infra-red Physical Layer Specification*, October 1998. Version 1.3.

58. Intanagonwiwat, C., et al. "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks." *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking*. August 2000.

59. Israeli Air Force, "IAF Aircraft," 2003. http://www.iaf.org.il/iaf/doa_iis.dll/Serve/item-/English/1.3.3.1.1.1.html.

60. Jantz, Scott D., et al. "Kinetics of Robotics: The Development of Universal Metrics in Robotic Swarms." *Proceedings of the 1997 Florida Conference on Recent Advances in Robotics*. April 1997.

61. Johnson, Paul, "Sensorcraft: Tomorrow's Eyes and Ears of the Warfighter." Briefing presented to the Association for Unmanned Vehicle Systems International, Wright-Kettering Chapter, May 2000. Briefing slides.

62. Joint Chiefs of Staff. *Joint Vision 2010*. Washington DC: Defense Technical Information Center, July 1996.

63. Jones, Richard M. *Introduction to MFC Programming with Visual C++*. Englewood Cliffs, NJ: Prentice Hall, 1999.

64. K-Team, "Kephera Robot." http://www.k-team.com/download/khepera.html, 2002.

65. Kadrovach, B. Anthony and Gary B. Lamont. "Design and Analysis of Swarm-based Sensor Systems." *Proceedings of the Midwest Symposium on Circuits and Systems*. August 2001.

66. Kadrovach, B. Anthony and Gary B. Lamont. "A Particle Swarm Model for Swarm-based Networked Sensor Systems." *Proceedings of the 2002 ACM Symposium on Applied Computing*. 918–924. Madrid, Spain: ACM Press, March 2002.

67. Kahn, J. M., et al. “Next Century Challenges: Mobile Networking for Smart Dust.” *Proceedings of the ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom 99)*. August 1999.

68. Kaufman, Gail, “Simulating Ants’ Behavior May Help U.S. Fight Future Wars,” February 2003. http://www.defensenews.com/pgt.php?htd=isr_1619235.html&tty=c4isr.

69. Kennedy, James and Russell C. Eberhardt. *Swarm Intelligence*. San Fransisco, CA: Morgan Kaufman, 2001.

70. Krock, Lexi, “Spies that Fly: Timeline of UAVs.” www.pbs.org/wgbh/nova/s;iesfly/uavs.html, November 2002.

71. Lab, Air Force Research. *Model Based Vision Laboratory*. http://www.mbvlab.wpafb.af.mil/ projects, June 1999.

72. Laboratory, UCLA Parallel Computing, “Global Mobile Information Systems Simulation Library,” February 2001. http://pcl.cs.ucla.edu/projects/glomosim/.

73. Li, Jinyang, et al. “A scalable location service for geographic ad hoc routing.” *Proceedings of the sixth annual international conference on Mobile computing and networking*. 120–130. ACM Press, 2000.

74. Logsdon, Tom. *Understanding the Navstar GPS, GIS, and IVHS* (Second Edition). New York, NY: Van Nostrom Reinhold, 1995.

75. Mackin, Joe, “Smart Sensor Web Integration.” Defense Science and Technology Seminar on Emerging Technologies, November 2000. Briefing slides.

76. Mano, M. Morris. *Digital Design*. New Jersey: Prentice Hall, 1984.

77. Mataric, Maja J. “Issues and approaches in the design of collective autonomous agents,” *Robotics and Autonomous Systems*, *16*(2–4):321–331 (December 1995).

78. Mataric, Maja J. “Coordination and Learning in Multi-Robot Systems,” *IEEE Intelligent Systems* (Mar/Apr 1998).

79. McClave, James T., et al. *Statistics for Business and Economics* (7th Edition). New Jersey: Prentice Hall, 1998.

80. The Microsoft Corp. *Microsoft Developers Network Library, Visual Studio*, 2000. v6.0.

81. The Microsoft Corp. *Microsoft Visual C++*, 2000. v6.0.

82. Morris, Robert, et al. “CarNet: A Scalable Ad Hoc Wireless Network System.” *Proceedings of the 9th ACM SIGOPS European workshop: Beyond the PC: New Challenges for the Operating System*. September 2000. The published version incorrectly lists Douglas De Couto’s name.

83. Murphy, D. W., et al. “MSSMP: No Place to Hide.” *Proceeding of the Conf. for the Assoc. for Unmanned Vehicle Systems Intl.*. June 1997.

84. Nyman, Jeff, "The Measure vs. The Metric." http://www.globaltester.com/sp6/measure.html, 2002.

85. O'Hara, Bob and Al Petrick. *The IEEE 802.11 Handbook: A Designer's Companion*. New York, NY: IEEE Press, 1999.

86. OPNET Technologies, Inc., 7255 Woodmont Ave., Bethesda, MD 20814. *Modeler Tutorial*, February 2002.

87. OPNET Technologies, Inc., 7255 Woodmont Ave., Bethesda, MD 20814. *ModelerXE Online Documentation*, February 2002. OPNET Modeler v8.0.C (build 1283).

88. OPNET Technologies, Inc., "OPNET Modeler–Accelerating Networking R&D," 2002. http://www.opnet.com/products/modeler/home.html.

89. Pachter, M. and P. R. Chandler. "Challenges of Autonomous Control," *IEEE Control Systems Magazine*, *18*(4):92–97 (August 1998).

90. Pages, Cover, "General SGML/XML Applications," 2002. http://xml.coverpages.org/gen-apps.html.

91. Partridge, B. L. "The Structure and Function of Fish Schools," *Scientific American*, *246*:114–123 (June 1982).

92. Passino, Kevin, et al. "Cooperative Control for Autonomous Air Vehicles." *Proceedings of the Cooperative Control and Optimization Workshop*. December 2000.

93. Pennebaker, William B. *JPEG Still Image Data Compression Standard*. New York: Van Nostrand Reinhold, 1993.

94. Perrone, Luiz Felipe, et al. "Modeling and Simulations Best Practices for Wireless Ad Hoc Networks." *Proceeding of the 2003 Winter Simulation Conference (WSC 2003)*. December 2003. Accepted.

95. Pike, John, "Low Cost Autonomous Attack System Miniature Munition Capability." Federation of American Scientists, Military Analysis Network, November 1999. www.fas.org/man/dod-101/sys/smart.

96. Pucknell, Douglas A. and Kamran Eshraghian. *Basic VLSI Design, Systems and Circuits*. New Jersey: Prentice Hall, 1988.

97. Rappaport, Theodore S. *Wireless Communications: Principles and Practice* (2nd Edition). New Jersey: Prentice Hall, 2001.

98. Reynolds, Craig. *Flock Applet: Boids in Java*. Reynolds Engineering and Design, 2000. http://www.red3d.com/cwr/boids/applet.

99. Reynolds, Craig W. "Flocks, Herds, and Schools: A Distributed Behavioral Model," *Computer Graphics*, *21*(4):25–34 (1987).

100. Riley, George and Alfred Park. *Parallel/Distributed NS*. Parallel and Distributed Simulation Lab, Georgia Institute of Technology, July 2003. http://www.cc.gatech.edu/computing/compass/pdns.

101. Royer, Elizabeth M. and Chai-Keong Toh. "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks," *IEEE Personal Communications*, *6*:46–55 (1999).

102. Samson, Victoria. *Q&A on the Use of Predator in Operation Enduring Freedom*. The Center for Defense Information, February 2002. http://www.cdi.org/terrorism/predator.cfm.

103. Sandia National Laboratories, Sandia Corporation. *Avalanche victims may be found four times faster*, Jan. 2000. http://www.sandia.gov/media/NewsRel/NR2000/-avalanch.htm.

104. Sandia National Laboratories, Sandia Corporation. *Mini-robot Research*, Jan. 2001. www.sandia.gov/media/NewsRel/NR2001/minirobot.htm.

105. Saravanan, N. and D. Fogel. "An Empirical Comparison of Methods for Correlated Mutations Under Self-adaptation." *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*. 479–485. Cambridge, MA:MIT Press, 1996.

106. Saunders, P.T., editor. *Morphogenesis: Collected Works of AM Turing*, *3*. North-Holland, 1992.

107. Schwamb, T. M., et al. "Performance Analysisi of a Dynamic Bandwidth Allocation Algorithm in a Circuit-Switched Communications Network." *Proceedings of the IEEE MILLCOM 20021*. 35–39. October 2002.

108. Sea Grant College Program, MIT. *Autonomous Underwater Vehicle Laboratory*, 2002. http://auvlab.mit.edu.

109. Skiena, Steven S. *The Algorithm Design Manual*. New York: Springer–Verlag, 1998.

110. Sklar, Bernard. *Digital Communications*. New Jersey: Prentice Hall, 1988.

111. SpaceDaily News. *Global Hawk UAV Completes 1,000 Hours Of Combat Support*, July 2002. http://www.spacedaily.com/news/uav-02q.html.

112. Stark, Henry and John W. Woods. *Prbability, Random Processes, and Estimation Theory for Engineers*. New Jersey: Prentice Hall, 1994.

113. Stevens, W. R. *TCP Illustrated*, *1*. Reading, MA: Addsion-Wesley, 1994.

114. Stressing, D. "The Beauty of Biomimicry," *Laptop*, 94–102 (October 2002).

115. Subramanian, Lakshminarayanan and Randy H. Katz. "An Architecture for Building Self-Configurable Systems." *IEEE/ACM Workshop on Mobile Ad-Hoc Networking and Computing*. August 2000.

116. Subramonian, V., et al. "Towards a Pattern Language for NEST Middleware." *Proc. of Object-Oriented Programming, Systems, Languages, and applications Workshop*. October 2001.

117. Sun Microsystems, "JINI Protocols," September 2000. http://developer.java.sun.com/devel-oper/technicalArticals/jini/protocols.html.

118. Tan, K.C., et al. "Autonomous robot navigation based on fuzzy sensor fusion and reinforcement learning." *Proceedings of the 2002 IEEE Internatinal Symposium on Intelligent Control*. 182–187. 2002.

119. Tanenbaum, Andrew S. *Computer Networks* (4th Edition). New Jersey: Prentice Hall, 2002.

120. Thomas, Ryan, et al. "Simulation, modeling, and evaluation of satellite-based multicasting protocols." *Proceedings of the 2001 Vehicular Technology Conference*. 2001.

121. Trahan, Michael W., et al. "Swarms of UAVs and Fighter Aircraft." *Proceedings of the 2nd Int'l Conference on Nonlinear Problems in Aviation and Aerospace*. 1998.

122. Univ. of Southern California. *The Information Sciences Institute*, 2002. http://www.isi.edu.

123. U.S. Air Force. *Air Force Issues Book*, 1997. Appendix B.

124. USAF Scientific Advisory Board. *Report on Building the Joint Battlespace Infosphere, Volume 1: Summary*, December 1999. SAB-TR-99-02.

125. von Eicken, T., et al. "Active Messages: a Mechanism for Integrated Communications and Computation." *Proc. of the 19th Int'l Symposium on Computer Architecture*. May 1992.

126. Warneke, B. A. and K. S. J. Pister. "MEMS for distributed wireless sensor networks." *Proceedings of the 9th Int'l Conf on Electronics, Circuits and Systems, Vol. 1*. 291–294. 2002.

127. Welch, Brent. *Practical programming in Tcl & Tk* (2nd Edition). Upper Saddle River, NJ: Prentice Hall PTR, 1997.

128. Werner, G. M. and M. G. Dyer. "Evolution of Herding Behavior in Artificial Animals." *From Animals to Animats 2: Proceedings of the 2nd Int'l Conf. on Simulation of Adaptive Behavior*. Cambridge, MA: MIT Press, 1992.

129. Williams, S. and I. Millar. "The IrDA Platform." *Proceedings of the 2nd Int'l Workshop on Mobile Multimedia Communications*. 1995.

130. Woo, Seung-Chul M. and Suresh Singh. "Scalable routing protocol for ad hoc networks," *Wireless Networks*, *7*(5):513–529 (2001).

131. Zeng, Xiang, et al. "GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks." *Workshop on Parallel and Distributed Simulation*. 154–161. 1998.

Vita

Major B. Anthony Kadrovach enlisted in the US Air Force and entered active duty in 1981. During the next six years he served as a Radiotelephone Operator (Russian) and Mission Supervisor, rising in rank from Airman Basic to Staff Sergeant. While on active duty he was accepted into the Airmen's Education and Commissioning Program (AECP). He attended Wright State University and received a Bachelor's Degree in Electrical Engineering in June 1990 whereupon he entered the Air Force Officer Training School (OTS).

After successful completion of OTS Tony was commissioned a 2nd Lieutenant, US Air Force Reserve in October 1990. He then served as Logistics Support Engineer for the C-5/Galaxy System Program Management Division. Tony was then selected to attended the Air Force Institute of Technology (AFIT) where he received a Master's Degree in Electrical Engineering in December 1995 and was a Distinguished Graduate. His follow-on assignment was to the Air Force Research Laboratory as a Microelectronics Engineer and System Test Leader. Before his current assignment to AFIT, he was assigned to the National Air Intelligence Center where he served as a Space Intelligence Engineer.

Tony's military assignments include Lackland AFB, TX; Goodfellow AFB, TX; Tempelhof AB, West Berlin, GE; Kelly AFB, TX; and Wright-Patterson AFB, OH (twice). His temporary duty assignments and travels have taken him to 27 states, Washington DC, England, France, Austria, and Spain. His AFIT follow-on assignment is with the Air Force Research Laboratory, Space Vehicles Directorate, Kirtland AFB, NM.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 074-0188*

1. REPORT DATE *(DD-MM-YYYY)* 09-03-2003	**2. REPORT TYPE** **Doctoral Dissertation**	**3. DATES COVERED** *(From – To)* Oct 1999 – Sep 2003

4. TITLE AND SUBTITLE A COMMUNICATIONS MODELING SYSTEM FOR SWARM-BASED SENSORS	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Kadrovach, Brian A. Major, USAF	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology; Graduate School of Engineering and Management (AFIT/EN); 2950 P Street, Building 640; WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/DS/ENG/03-03
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/IFTA; Attn: Dr. Robert L. Ewing; 2241 Avionics Cl; WPAFB OH 45433; DSN: 785-6653x3592; e-mail: Robert.Ewing@wpafb.af.mil	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Today's information age has exploded the amount of data available to decision makers of all types. The miniaturization and proliferation of sensor technology has enabled extensive detection and monitoring, and advances in computational capabilities have provided for embedded data analysis and the generation of information from raw data. Additionally, with the miniaturization of mechanical systems, it is possible to provide platforms for sensor suites that are capable of mobility and limited autonomy. Swarming, or bio-emergent behavior, provides a robust, scalable mechanism for organizing large numbers of mobile sensor platforms. However, the mobility dynamics of swarm systems present additional challenges.

This research develops a novel ad hoc data network communications modeling methodology for swarm-based sensor systems that provides a process for evaluating performance of communications protocols with respect to swarm dynamics. A new parameter-based swarm simulation system based on innovative vision models is developed and used to investigate and characterize swarm behavior. The process allows for communications protocol evaluations in the context of dynamic swarm behaviors.

Three network communications protocols are presented for swarm based sensor networks and a performance comparison is made. The three protocols—Directed Diffusion, Geographical Routing Protocol, and Flooding Protocol—are compared. Results indicate, for the degree of mobility investigated, that the Directed Diffusion protocol outperforms the Geographical Routing Protocol system. The swarm network modeling process developed provides a new methodology for rigorous and repeatable investigation of network communications systems with respect to the complex dynamics of swarm based sensor networks.

15. SUBJECT TERMS

Behavior, animal communication, computerized simulation, flow visualization, algorithms, network topology, communications networks, sensor fusion, robotics

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Gary B. Lamont, AD-24, USAF (ENG)
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636, ext 4718; e-mail: Gary.Lamont@afit.edu
U	U	U	UU	252	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18