

Geographic Based Ad-Hoc Routing for Distributed Sensor Networks

A Thesis Submitted to the Faculty
in partial fulfillment of the requirements for the
degree of

Master of Science

by

Michael G. Corr

Thayer School of Engineering
Dartmouth College
Hanover, New Hampshire

June 1, 2001

Examining Committee:

Clayton M. Okino, Ph.D., Chair

George V. Cybenko, Ph.D, Member

Robert S. Gray, Ph.D., Member

Dean of Graduate Studies

(Signature of Author)

Abstract

Recent advances in sensing devices and integrated circuit technology have allowed for the development of easily “reconfigurable smart sensor” products for the application of Distributed Sensor Networks. Primarily utilizing commercial off-the-shelf (COTS) components, I have developed reconfigurable smart sensors, consisting of a microprocessor, GPS receiver, RF transceiver, and temperature sensor. The result is a flexible module capable of gathering sensor data and forwarding compressed information to a central location via other modules.

In this Thesis, I will present a new paradigm with relaxed requirements for ad hoc sensor network routing protocols called *Statistically Accurate Sensor Networks* (SASN). Using the guidelines of a SASN, an innovative self-routing algorithm called *Best Effort multi-Hop Geographical Routing* (BEHGR), will be proposed for the purpose of networking wireless sensors. In BEHGR, geographical positioning information is utilized to dynamically route packets to a central location in a “best effort” manner. Using the reconfigurable smart sensors as a real time test bed, I have deployed and tested the BEHGR protocol in the field. The infrastructure design for these sensor modules, as well as performance measurements for data throughput and data currentness of the BEHGR protocol are presented and analyzed.

Contents

1	Introduction and Overview	1
1.1	Sensor Network Routing	2
1.2	Statistically Accurate Sensor Networks	3
2	Connectionless Oriented Location Aware Ad Hoc Sensor Routing	5
2.1	Location Awareness	6
2.2	Route Discovery Algorithm	10
2.2.1	Initial State Calculations	13
2.2.2	Dynamic State Updating	16
2.2.3	Direction Checking	18
2.3	Other Aspects of Ad Hoc Routing	18
2.3.1	Route Maintenance	18
2.3.2	Fairness and Load Balancing	19
2.3.3	Mobility	19
2.3.4	Preventing Loops	21
2.3.5	Multiple Home Units	22
2.4	Performance Metrics	23
2.4.1	Throughput	23
2.4.2	Currentness	26
2.5	Numerical Results	26
2.5.1	Testing Topology	27
2.5.2	Throughput	27
2.5.3	Currentness	32
2.5.4	Average Route Time	34
2.5.5	Load Balancing and Performance	35
2.6	Discussion	36
3	Conclusion	38
A	Background Philosophy of the BEHGR Protocol	42
B	Sensor Module Test Bed Hardware Design	44
B.0.1	Hardware Design	45
B.1	Hardware	47
B.1.1	Microprocessor	47
B.1.2	Radio Transceiver	47

B.1.3	Global Positioning System	48
B.1.4	Sensor	49
C	Connection Oriented Location Aware Ad Hoc Sensor Protocol ¹	50
C.1	“Who’s Out There?” cost function (WOT-CF) algorithm	52
C.1.1	Cost Function Example	55
C.2	Simulation Results	58
C.2.1	Route Link Setup	59
C.2.2	Routability and Network Density	59

List of Figures

2.1	Raw TTFL GPS readings	8
2.2	Probability Distribution for TTFL	8
2.3	Probability Distribution for TTFL per Number of Locked Satellites	10
2.4	Flow diagram of BEHGR state changes	12
2.5	Client Mode Time - The Client Mode time calculations and adaptation process is viewed as a Markov Chain consisting of transition probabilities for each oscillation period, where M_{max} is the maximum allowable value for \hat{C} . For state i , p_i is the probability of not locking, q_i is the probability of locking, but not receiving a valid tier, and $1 - p_i - q_i$ is the probability of locking and receiving a valid tier value.	16
2.6	Client - Server interaction for Client state duration calculation	17
2.7	A sample network topology used for throughput testing	27
2.8	Data throughput vs. number of nodes	29
2.9	Home location testing network topologies	29
2.10	Data throughput vs. position of Home	30
2.11	Percentage of data throughput per module	31
2.12	Data throughput as queue size changes	32
2.13	Currentness and Average Route times vs. queue size	33
2.14	Currentness versus position	34
2.15	Currentness per Home position	34
2.16	Load balancing in a network topology	36
B.1	Two sensor designs exploiting modularity	45
B.2	A sensor design exploiting modularity	46
C.1	Cost Function Value Algorithm	56
C.2	Geographic Address and Routing Protocol	57
C.3	Average time to route per node	60
C.4	Percent of nodes routable	61

Chapter 1

Introduction and Overview

Recent advances in sensing devices and integrated circuit technology have helped promote the interest in research and development of ad hoc networks. Specifically, research in the area of routing protocols for ad hoc networks [14] [17], has recently seen considerable progress. A subset of ad hoc networks is the application of sensor networks, where the primary objective of a sensor network is to collect and forward sensing data.

Geographical routing methodologies utilizing GPS have been presented for position identification [10] [12] [13] [3] [1]. Extending on this concept, [6] utilizes position information explicitly as the sensor identifier as opposed to traditional IP labeling or the wireless equivalent of layer 1 external gateway routing protocols such as BGP-4 [8], thus eliminating the need to append position information in the data payload. Moreover, in a dynamic environment, minimal interaction of link connection status

while maintaining relative position within the network may be sufficient for statistically forwarding information towards a central location.

1.1 Sensor Network Routing

In a typical ad hoc sensor network, the final destination of a packet is located outside the physical transmission range of the sender. For this reason, a sender must locate a viable link through which the packet can be sent. A viable link consists of another sensor module capable of receiving and forwarding the message packet. This process continues in a hop by hop manner until the message packet reaches the final intended destination. Message hopping has been proven effective for propagating information throughout a widely dispersed wireless ad hoc network.

Routing protocols determine the means through which these hops are discovered and maintained. There are currently two general classifications of protocols, table-driven, and on-demand[17]. Once routes are established, in a table-driven routing protocol, the routes are constantly maintained with up to date routing information. While this requires more maintenance overhead, pre-established connections allow for immediate forwarding of information from module to module.

On demand vector routing is more commonly seen in ad hoc wireless networks, as it allows more flexibility. In on demand routing protocols such as Ad hoc On Demand Distance Vector (AODV) Routing [15], routes are not established until a module initiates communication with another module in the network. The route

discovery process entails a series of request messages sent between neighbor modules, demanding a new route. Once established, each route is maintained for the duration of the conversation. When the original module (source module) is finished with the conversation, the route is broken and no longer maintained.

On a similar note, for sensors distributed such that there are N sets of n nodes where each node is capable of unidirectional transmission based on a power level of radius d_{rf} over a unit circle area, Gupta and Kumar [7] have shown that given each node covers an RF circular area of $\pi d_{rf}^2 = \frac{\log n + c(n)}{n}$, then the network approaches connectivity of probability 1.

The AODV protocol and the connectivity analysis of Gupta and Kumar may in fact be stronger requirements than necessary for sensor networks.

1.2 Statistically Accurate Sensor Networks

An aspect of sensor networks that has yet to be leveraged is the concept of aggregate acceptable performance. Specifically, not all information among all the sensors is required to reach a central location for a network to be considered operational. We present the concept of Statistically Accurate Sensor Networks (SASNs). In a SASN, an overall collection of sensor information from a distribution of sensors is statistically sufficient for representing the current state of the sensor network.

As ad hoc sensor networks increase in size and density, complete routing and connectivity become intractable. In very large scale networks, such as the proposed

network for Smart Dust [18], not all sensor particles are expected to route. The concept of only a subset of the particles routing in a very large scale network lends itself to the SASN philosophy. Essentially, a SASN relaxes the requirements of an ad hoc sensor network routing protocol.

In this Thesis we propose and analyze an innovative routing protocol for the SASN paradigm that relies only on minimal location and current local state information. The protocol, called *Best Effort multi-Hop Geographical Routing* (BEHGR) ¹ is a connectionless oriented location aware routing methodology that attempts to sufficiently collect and route distributed sensor information to a central location.

The BEHGR algorithm is designed to handle network connectivity without the requirement of a high level MAC layer such as the *IEEE 802.11* standard. This design was specifically chosen for low power radio communication devices such as OEM transceivers operating in *Industrial, Scientific, and Medical* (ISM) bands, or the recent *Bluetooth*TM standard. The inherent Client-Server relationship found in the *Bluetooth*TM protocol lends itself well to the BEHGR protocol, as will become more apparent in the following chapter.

¹The words “best effort” are in the spirit of the IP protocol with respect to attempted quality of service. The motivation to provide guaranteed service in traditional wire line and wireless networks has led to restricted requirements in ad hoc networking.

Chapter 2

Connectionless Oriented Location Aware Ad Hoc Sensor Routing

Once deployed, modules in a sensor network generally have one task: collect and send data to a central location. In order to fulfill this obligation, sensor modules need to follow an algorithm or protocol for optimizing hop count, data throughput, network efficiency and battery consumption.

In this and subsequent sections, we present BEHGR (*Best Effort multi-Hop Geographic Routing*), the routing protocol used for our sensor network.¹ This protocol is non-conventional as no link state information is retained nor is an addressing scheme used. The protocol is connectionless oriented in that no link connection nor link state information is retained after a link is established or removed. Each time a module

¹A test bed of physical sensor modules was designed and built for empirical testing of the BEHGR algorithm. The design philosophy and hardware description can be found in Appendix B

wants to forward data packets, a separate, new link must first be created.

An underlying concept of this routing protocol is to exploit sensor location awareness information. Through the added information of location awareness, sensor modules can evaluate routes for effectively sending data packets to a destination. Specifically, the combination of location awareness and route discovery characterize the BEHGR algorithm.

2.1 Location Awareness

In a randomly distributed network as in the case of ad hoc sensor networks, location awareness is essential to establishing the routing topology. Location awareness of sensor modules distributed throughout the network requires the acquisition of information about each sensor's location with respect to an absolute or relative reference. In dynamic networks, the added information of the module's current location can greatly increase performance for route selection and maintenance.

Gaining popularity as a method for creating a location aware network is the use of the Global Positioning System (GPS). GPS is a global system consisting of a constellation of satellites constantly relaying information while orbiting the Earth. Utilizing the concept of position *triangulation*, receivers on Earth collect the signals transmitted by these satellites and calculate the exact location of the receiver on an absolute global scale. The resulting calculated position is in reference to the Latitude and Longitude coordinate system.

The commercial market has seen a lot of development in GPS receivers, causing an increase in functionality and a decrease in pricing. As a result, embedding a GPS receiver into every sensor module is a plausible idea.

Although it may be economically feasible to embed every sensor module with a GPS receiver, it has not been determined if a high enough level of reliability is attainable when using GPS receivers in a dynamic sensor network. For this reason, preliminary studies of GPS performance are necessary. The primary concern is the amount of time required to acquire an accurate position. If performance of the receiver is not acceptable, functionality of the sensor network may be adversely affected.

An initial study was performed on a particular make and model GPS receiver.² The study included calculations of expected time distribution curves for a Time To First Lock (TTFL). A TTFL is defined as the amount of time required by the GPS receiver to acquire a position reading within 10 meters³ of the actual receiver position. Note that this TTFL metric differs from the more common performance parameter, Time To First Fix (TTFF) in that the first “fix” may not be sufficiently accurate for our purposes.

To find the TTFL distribution curve, a single GPS receiver was placed in a fixed and known position. Controlling software then continuously turned the receiver on, waited for a TTFL, recorded the time, then turned the receiver off for a brief period of time. This cycle was repeated for a continuous duration of 2 days. Figure 2.1

²The GPS receiver was a GT+ Oncore model made by Motorola.

³10 meters is an acceptable resolution for the GPS system.

shows the raw TTFL times throughout the testing period, while Figure 2.2 shows the histogram distribution curves for the collected times.

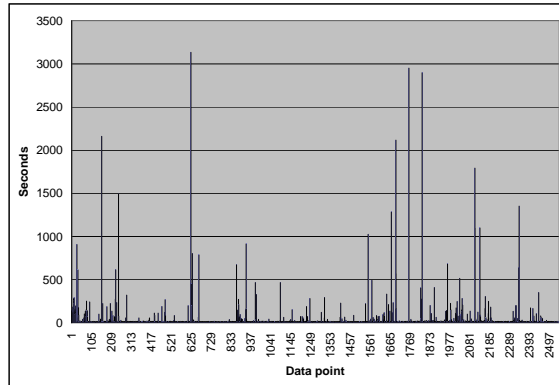


Figure 2.1: Raw TTFL GPS readings

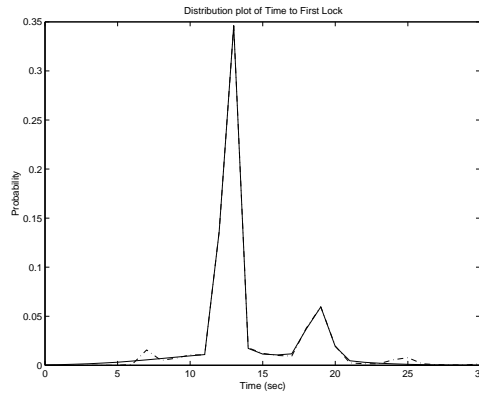


Figure 2.2: Probability Distribution for TTFL

Based on the collected GPS receiver data, the GPS TTFL appears to have a trimodal Gaussian distribution with probability density function,

$$f(i) = \frac{\phi_1}{\sqrt{2\pi}\sigma_1} \cdot e^{-\frac{1}{2}\left(\frac{i-\mu_1}{\sigma_1}\right)^2} + \frac{\phi_2}{\sqrt{2\pi}\sigma_2} \cdot e^{-\frac{1}{2}\left(\frac{i-\mu_2}{\sigma_2}\right)^2} + \frac{\phi_3}{\sqrt{2\pi}\sigma_3} \cdot e^{-\frac{1}{2}\left(\frac{i-\mu_3}{\sigma_3}\right)^2} \quad (2.1)$$

where σ_x is the variance, μ_x is the mean, and ϕ_x is a scaling factor for each individual Gaussian distribution.

Statistical analysis was performed, in an attempt to prove the trimodal Gaussian curve to be an accurate representation of the GPS distribution curve. Preliminary attempts could not provide a valid confidence metric using a Goodness-of-Fit test. We premise that more data points or an unbalanced, unsigned error metric would lead to proper validation of the trimodal distribution. Regardless, the curve is still reasonably accurate for predicting Time to First Lock times, and can be used effectively for modeling a distribution of acquisition times for sensor module locations.

Since there is a high probability of a TTFL time occurring at 13 seconds, this was deemed acceptable for adequate performance in terms of expected acquisition time. As a statistical upper bound, the majority of TTFL times fell within 30 seconds. Specifically, we can say that the TTFL is within 30 seconds with probability 0.9113.

Note, the number of visible satellites does not appear to greatly affect the TTFL distribution. A previous misconception was, the greater the number of visible satellites the faster a TTFL will occur. Figure 2.3 shows the TTFL distribution separated by the number of locked satellites at the time of the TTFL. The figure shows the distribution is ubiquitous for all number of locked satellites. Also note, the multimodal characteristics of the distributions are consistent among the number of satellites, showing the multimodal nature is not caused by a variance in the number of satellites used to acquire a TTFL.

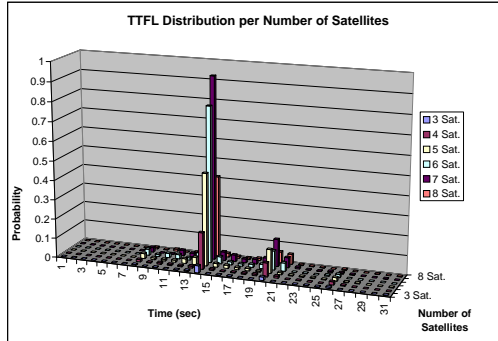


Figure 2.3: Probability Distribution for TTFL per Number of Locked Satellites

2.2 Route Discovery Algorithm

The route discovery process in BEHGR is unique to ad hoc networking inclusive of both sensor networks and communication networks. In the algorithm, each module is in one of two states, *Server* or *Client* and can only communicate with another module in the reciprocal state. Data packets flow in the direction of Client to Server, while control packets flow in the reverse. In order to propagate information from one part of the network to another, each module must oscillate between Server and Client states. We define an *oscillation period* as the amount of time for a single loop through Client and Server states. By oscillating between the two states, two neighboring modules will simultaneously *lock* together when each module is in a complementary state to each other. The duration a module remains in each state is derived from the module's radial distance to a central location point, called the *Home* unit. All sensor modules must send their collected data to one or more of these Home units⁴. In many cases,

⁴It is possible for a sensor network to have multiple Home units for the purpose of data collection

this requires utilizing neighboring modules in a hop by hop manner to forward data to the Home location. Knowledge of the location of the nearest Home is required for evaluating a current route. Once a Client and Server lock to each other, location information is exchanged from the Server to Client. If the Server's position is in the direction towards Home, then the lock is valid, else the connection is ignored.

In the initial state of the network, no module has knowledge of the location of the Home unit. The acquisition of Home's location begins with each module in a Client state continuously monitoring the RF channel until a lock with a Server occurs. Once a lock with a Server is established, the Client sends a *MSG_ReqHome* message to the Server requesting the location of Home. Note that a module will never enter a Server state unless the module has previously acquired Home's location information. After the module in Client state receives the location of Home, the module can begin an oscillation period as depicted in Figure 2.4. Once a module has begun to oscillate, as a Server, the module may lock to any neighboring module in Client state, including modules still requesting Home's location. The module in Server state will acknowledge the request with the proper information for this and all possible subsequent requests. As each module acquires an initial lock, the Home unit's location information propagates outwards in a radial manner from the Home unit.

The duration of time spent in each of the two states varies based on the immediate history of each module. Note that a system that did not allow for dynamic time

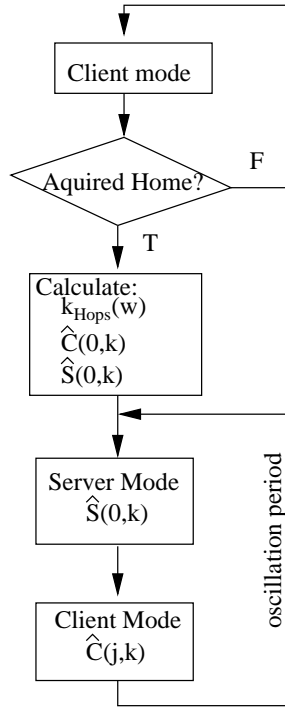


Figure 2.4: Flow diagram of BEHGR state changes

duration could result in two neighboring modules never synchronizing. This scenario could impede the throughput of packets in the network.

Since all modules begin in Client state, there must be some initial unit in a Server state with Home's location information. This initial module is the Home unit itself. The network will not begin its route discovery process until the Home unit has disseminated its location information to the network. After the initial lock, each module is free to oscillate, with the goal of locking each oscillation period to an appropriate Client or Server.

2.2.1 Initial State Calculations

A Hop is defined as the maximum radial distance between two nodes for proper RF communication. This value is typically set by the propagation distance of an RF transceiver and is selected to ensure an acceptable bit error rate (BER), typically 10^{-6} or better. In order to estimate the number of hops a module will require to forward a data packet home, the radial distance from the module to home must first be calculated. We define $D_{radial}(w)$ as the radial distance from module w to the Home unit. Mapping to a Cartesian coordinate system, with Home's position at (\hat{x}, \hat{y}) , we can write $D_{radial}(w)$ as,

$$D_{radial}(w) = \sqrt{(x_w - \hat{x})^2 + (y_w - \hat{y})^2} .$$

We define the *hop metric* as the estimated number of hops required to transmit a packet from a given module, w , to Home. Specifically, the hop metric for unit w is,

$$k_{Hops}(w) = D_{radial}(w)/d_{rf} , \tag{2.2}$$

where d_{rf} is the selected propagation length for the RF transceiver. In terms of the hop metric, we can write the set of modules belonging to the k^{th} tier, also referred

to as the k^{th} tier set as,

$$\hat{N}(k) = \{w : \lfloor k_{Hops}(w) \rfloor = k \quad \forall w\} .$$

We define the initial Client state duration, $\hat{C}(0, k)$, and initial Server state duration, $\hat{S}(0, k)$, as functions of the k^{th} tier set. We write these as

$$\hat{C}(0, k) = H_{max} + k,$$

and

$$\hat{S}(0, k) = H_{max} - k,$$

where H_{max} is the maximum allowed number of hops from the outermost sensor module in the network to Home and

$$k = \{w : \lfloor k_{Hops}(w) \rfloor \quad \forall w \in \hat{N}(w)\},$$

such that $\lfloor x \rfloor$ is the floor function. For all subsequent oscillation periods j , $\hat{C}(j, k)$ and $\hat{S}(j, k)$ are defined later in Section 2.2.2.

Recognize that a module can only send data packets when it is in Client state. Modules closer to Home will be responsible for receiving more packets than modules positioned closer to the network's outer perimeter. Thus, modules closer to Home shall have longer $\hat{S}(0, k)$ cycle times than their corresponding $\hat{C}(0, k)$ cycle time, i.e.

we can write

$$\hat{S}(0, k) > \hat{S}(0, k + 1) \quad \forall k .$$

Notice in Figure 2.4, a calculated $\hat{S}(0, k)$ remains fixed. The ultimate goal of each module w at tier k , is to synchronize with at least one module, w' , with a lower tier value, $k - 1$ (i.e. module w' is closer to Home than module w). True synchronization occurs when a module at tier k locks to a correct Client at $k + 1$ and Server at $k - 1$ for each oscillation period. When all modules are synchronized to each other, the maximum data throughput can be achieved. In order to achieve synchronization, there must be some fixed, steady frequency in the network. This explains the fixed duration of the Server state. The Client is responsible for adjusting the duration value of $\hat{C}(j, k)$ in order to synchronize to a Server with duration $\hat{S}(j, k - 1)$ and not the reciprocal.

If the Server duration time value was allowed to change simultaneously with the Client time value, unstable oscillation effects could occur, as was observed in preliminary testing.

We can then write, for all j , the Server duration as,

$$\hat{S}(j, k) = \hat{S}(0, k) = H_{max} - k,$$

for each module w .

The adaptation of locking times for all modules attempting synchronization orig-

inates with a steady state oscillation frequency from Home. Home has a fixed steady state oscillation frequency, to which all other modules will eventually synchronize, either directly or indirectly. In particular, Home's oscillation period differs from all other modules' in that Home does not need to enter the Client state, i.e. Home does not transmit data packets to other modules, only control packets. Thus, Home simply oscillates between acquiring locks and receiving data packets, while maintaining Server mode.⁵

2.2.2 Dynamic State Updating

After the initial setup, a module may either increase, decrease, or hold the $\hat{C}(j, k)$ duration, depending on the result of the module's previous oscillation period.

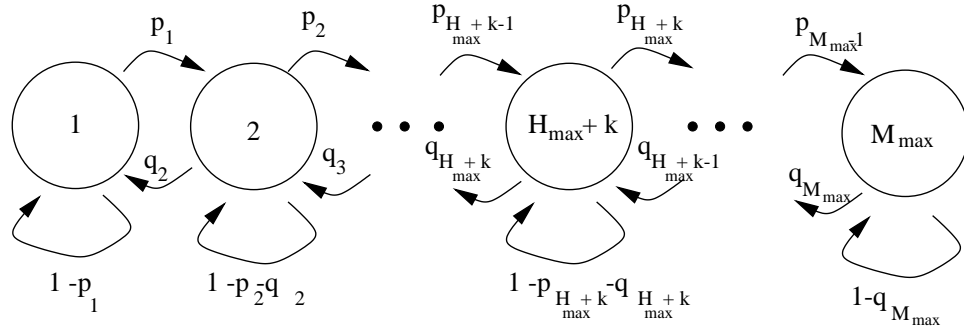


Figure 2.5: Client Mode Time - The Client Mode time calculations and adaptation process is viewed as a Markov Chain consisting of transition probabilities for each oscillation period, where M_{\max} is the maximum allowable value for \hat{C} . For state i , p_i is the probability of not locking, q_i is the probability of locking, but not receiving a valid tier, and $1-p_i-q_i$ is the probability of locking and receiving a valid tier value.

⁵Note, in the current implementation, a module cannot be acquiring locks and receiving or sending RF message packets concurrently. Each state, Client and Server, has two sub-states, *Acquisition* and *Communication* which are independent of each other.

As depicted in the Markov chain found in Figure 2.5, if a module is in the Client state and does not find a Server to lock with, then the module will increase the $\hat{C}(j, k)$ time by a value of T_{step} , where $T_{step} = 1$.

If a module, while in Client state, does lock with a Server but after negotiation determines that the Server is not in the direction of Home, then $\hat{C}(j, k)$ will be decreased by a value of T_{step} .

A module will hold its value of $\hat{C}(j, k)$ if it locks with a Server and it determines the Server module is positioned correctly, in the direction towards Home. We can write the time duration as a Client with respect to Figure 2.6 as,

$$\hat{C}(j, k) = \begin{cases} \hat{C}(j - 1, k) - T_{step} & \text{if Client } w \text{ locks with Server } w', \text{ but does not} \\ & \text{receive a valid tier value, } k < k' \\ \hat{C}(j - 1, k) & \text{if Client } w \text{ locks with Server } w' \text{ and} \\ & \text{receives a valid tier value, } k \geq k' \\ \hat{C}(j - 1, k) + T_{step} & \text{if Client } w \text{ does not lock at all, } k' \text{ unknown} \end{cases} \quad (2.3)$$

where j is the current round of a Client evaluating and negotiating a lock with a Server. For an in depth philosophy of the adaptation process see Appendix A. Detailed analysis of the Markov chain in Figure 2.5 is beyond the scope of this work.

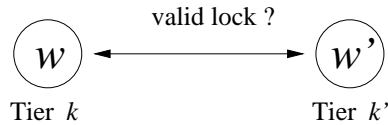


Figure 2.6: Client - Server interaction for Client state duration calculation

2.2.3 Direction Checking

Once a module w in Client state and a module w' in Server state lock with each other, the Client will first determine if the lock with Server w' is in fact a lock in the direction towards the final destination, i.e. the Client determines a valid lock. A Server, once locked to the Client, immediately sends a *MSG_TierACK* message containing the Server's tier value, k' . The Client's tier value, k is then compared to the received value, k' , to determine if a valid connection ($k > k'$), exists. Once a valid connection has been established, the Client will send data packets to the Server within the time duration of $\hat{C}(j, k)$.

2.3 Other Aspects of Ad Hoc Routing

2.3.1 Route Maintenance

In the BEHGR protocol there is no typical route maintenance. A route is maintained when a module's Client state time duration, $\hat{C}(j, k)$, is at steady state and tuned such that the Client will lock to a correct Server every oscillation period. There is no link state information nor Server identity information retained by the module. The only information each module retains about the network is the Home unit's location and the outcome of the last Client state. No routing tables nor link route information is retained.

2.3.2 Fairness and Load Balancing

Using the BEHGR protocol, each sensor's data queue serves in a First In First Out (FIFO) manner with the possibility of overwrites, i.e. when the queue reaches the maximum size and loops back to the beginning. Ideally, a fair queuing system would allow for all sensors to serve each data packet in an equal manner. Because BEHGR allows the data queue to overflow and thus overwrite data, this cannot be considered a fair queuing system. Future investigations are required in allowing for true fairness for BEHGR as a function of distributed hop by hop queuing.

Load balancing attempts to achieve optimal use of all sensors in a balanced network topology. Using the BEHGR algorithm, load balancing can be achieved with an evenly distributed network of sensor modules. When the network is synchronized, there is a possibility for Clients to lock during each oscillation period to different Servers on a given tier. By doing so, the Client will randomly and evenly distribute data packets to these multiple Servers, thereby evenly distributing the workload.

2.3.3 Mobility

Within a sensor network, there are three possible scenarios for mobility.

1. The Home unit is stationary while the sensor modules are mobile.
2. The Home unit is mobile while the sensor modules are stationary.
3. Both the Home unit and sensor modules are mobile.

The BEHGR algorithm can adapt to all three situations. The first scenario is the most simple. As described above, only the module's own state information is preserved. This allows modules to freely change their location without disrupting the performance of the network. In an evenly distributed network, if a module w at tier k moves to a new location, all modules at $k + 1$, previously synchronized to module w will shift their synchronization patterns to another module on tier k . Ideally, these modules at tier $k + 1$ would already statistically be switching synchronization locks evenly among multiple modules in tier k , resulting in a minimal affect on the network due to module w 's departure.

When a module w moves to a new location, module w 's position information and time values: $\hat{C}(j, k)$ and $\hat{S}(0, k)$ are updated accordingly. This will allow module w to quickly synchronize to the new surrounding neighbor modules.

The second scenario, with the Home unit being mobile is also simple, but will require more time for the network to re-synchronize. In this case, when the Home unit moves, the Home unit's new location must propagate to all modules in the network. To do so, the new location information is appended to the *MSG_TierACK* message already being sent every Server state cycle of the Home unit and all sensor modules. When a Client module receives this message, the Client will extract the k tier information as customary, but will also parse the additional information of Home's new location. The Client module then updates the state information and appends the revised Home location information to a *MSG_TierACK* message for successive

transmissions. Eventually, the new location of the Home unit propagates through the network.

When both the Home unit and sensor modules are mobile, a combination of the above two mentioned procedures occur. For proper functionality, it is imperative that the rate at which Home's location information spreads through the network is proportional to the maximum of the velocity of the Home unit and the GPS tracking location rate. In a highly mobile network, there is a possibility for data from sensor modules to "chase" the Home unit. For this reason the queue size for data packets must be optimized as these packets will tend to move back and forth between modules until they reach Home. A larger queue size will allow the packets to live longer until the network has a chance to re-synchronize and reach steady state.

2.3.4 Preventing Loops

A route loop occurs when links between modules form a loop back to an origin, causing data packets sent on a route path to never reach the intended final destination. These loops tend to occur when modules do not have updated network state information, causing incorrect route links to be formed. There are several factors present in the BEHGR protocol to help prevent the formation of route loops. First, since each link is re-established every oscillation cycle, a non-updated or "stale" link cannot be formed.

Irrepressibly, BEHGR allows for packets to travel in the wrong direction. If Home is mobile and the new Home location information has not reached one or more mod-

ules, data packets could travel in the wrong direction. The protocol ensures that either the packets will be dropped once they reach a module with an updated Home location information, or the packets will be diverted and turned around towards the new Home position. Note that packet drops are acceptable because a Statistically Accurate Sensor Network (SASN) expects some acceptable loss such that not all data packets will reach Home. If the packets are diverted, they will have a longer route time, but again, in a SASN, not all packets are expected to be real-time, just sufficiently time current. Note, data packets are also permitted to reach home out of time sequence.

2.3.5 Multiple Home Units

If there is more than one Home unit in a network topology, each sensor module can react in one of two ways. Either the module will select the most recent Home location information the module receives for all calculations and data packet forwarding, or the module will store all Home unit location information and select the optimal one to which to send all data to. The former case is simpler to implement, but may not be the most reliable nor stable solution. If each module is empowered with the decision of which Home unit is optimal with respect to the module, a more powerful yet complex network may form. Further research is necessary to characterize the behavior and performance for each possible solution.

2.4 Performance Metrics

2.4.1 Throughput

Each module contains a data inflow rate and outflow rate. A combination of relative distance from Home and the lock history for $\hat{C}(j, k)$, will determine the total data flow rate and data queue backlog.

Once a Client and Server lock with each other, the Client will randomly select a number of packets to transmit within the time duration of $\hat{C}(0, k)$. Therefore, on average, each Client will transmit a total of $\frac{\hat{C}(0, k)}{2}$ packets in a single oscillation period, i.e. the expected rate of packets sent for module w in round j is $E[N_{p_sent}(j, w)] = \frac{\hat{C}(0, k)}{2}$, where $N_{p_sent}(j, w)$ is the number of packets sent by module w in round j .

We define the oscillation period as

$$T(j, k) = \hat{C}(j, k) + \hat{S}(0, k) ,$$

where the current oscillation period count is j and the current tier level is k .

For the k^{th} tier sensor module in oscillation period j , we can then define the instantaneous data outflow rate to be

$$r_{out}[j, k, w] = \frac{N_{p_sent}(j, w)}{T(j, k)} . \quad (2.4)$$

We can also define the instantaneous data inflow rate as the rate at which packets

are generated plus the rate at which packets are received. The rate at which packets are generated for module w is shown as

$$r_{gen}(w) = \frac{N_{p_gen}(j, w)}{T(j, k)} ,$$

where $N_{p_gen}(j, w)$ equals the number of generated packets by module w at round j . Now suppose module w is at tier k , let $\hat{N}(k+1)$ be the set of modules at tier $k+1$. Let $N(k+1)$ be the number of modules in the set $\hat{N}(k+1)$. Assuming a Server module in tier k only receives packets from a Client module in tier $k+1$, the maximum number of packets that can be received in a single oscillation period can be shown to be

$$N_{p_rcvd}(w) = N(k+1) \cdot \sum_{x=1}^{N(k+1)} r_{out}[j, k+1, x] ,$$

where $r_{out}[j, k+1, x]$ is the instantaneous data outflow rate for each module $x \in \hat{N}(k+1)$. Note, a single Server may receive data packets from multiple Clients within a single $\hat{S}(0, k)$ duration period.

The backlog for module w at round j is defined as the total inflow rate of data packets minus the total outflow rate of packets. This is shown as

$$B_w(j) = \sum_{i=1}^j (r_{in}[i, k, w] - r_{out}[i, k, w]) . \quad (2.5)$$

Theorem 1 *The data queue backlog for module w of the k^{th} tier, at the end of time slot j is*

$$B_w(j) = \sum_{i=1}^j \frac{1}{T(j, k)} \left[N_{p_gen}(j, w) + \left(N(k+1) \cdot \sum_{x=1}^{N(k+1)} N_{p_sent}(j, x) \right) - N_{p_sent}(j, w) \right].$$

Proof of Theorem 1: The data packet inflow rate is the summation of the number of packets generated by module w and the number of packets received by module w , in round j . This is shown as,

$$\begin{aligned} r_{in}[j, k, w] &= \frac{1}{T(j, k)} [N_{p_gen}(j, w) + N_{p_rcvd}(j, w)] \\ &= \frac{1}{T(j, k)} \left[N_{p_gen}(j, w) + \left(N(k+1) \cdot \sum_{x=1}^{N(k+1)} r_{out}[j, k+1, x] \right) \right] \\ &= \frac{1}{T(j, k)} \left[N_{p_gen}(j, w) + \left(N(k+1) \cdot \sum_{x=1}^{N(k+1)} N_{p_sent}(j, x) \right) \right], \quad (2.6) \end{aligned}$$

where $x \in \hat{N}(k+1)$.

Thus, replacing (2.5) with (2.4) and (2.6), we can define the data queue backlog of the k^{th} tier sensor module at the end of time slot i , as

$$\begin{aligned} B_w(j) &= \sum_{i=1}^j (r_{in}[i, k] - r_{out}[i, k]) \\ &= \sum_{i=1}^j \frac{1}{T(j, k)} \left[N_{p_gen}(j, w) + \left(N(k+1) \cdot \sum_{x=1}^{N(k+1)} N_{p_sent}(j, x) \right) - N_{p_sent}(j, w) \right]. \end{aligned}$$

□

2.4.2 Currentness

We propose a new metric of performance for distributed ad hoc sensor networks called *currentness*. The actual concept of currentness is not new, and in fact is defined in [5] for estimating the speed of re-indexing to maintain a level of currency focusing on web page access. We refine the definition of currentness for Distributed Sensor Networks (DSNs).

Definition 2 ((α, β)-currency of a source in a network) *A source in a network is said to be (α, β) current if a piece of generated information at time t_o is received at the final destination within some interval of time β with probability α , i.e. the piece of information is received before or at time $t_o + \beta$ with probability α .*

Similarly, we can present the concept of an entire network being (α, β) current with respect to a specific destination.

Definition 3 ((α, β)-currency of a network) *A network is said to be (α, β) current if all network generated information is received at a specific destination within some interval of time β with probability α .*

2.5 Numerical Results

In order to test the functionality and determine the characteristics of the BEHGR algorithm, empirical field testing was performed. Using the physical sensor mod-

ules described in Appendix B, several network topologies were set up to test data throughput, data currentness, and algorithm performance.

2.5.1 Testing Topology

In a typical ad hoc sensor network, sensor modules are randomly located throughout the network area. For testing purposes, several network topologies were specifically selected in order to provide a controlled testing environment. All network topologies were of a grid format configuration of $m \times m$ modules with a distance spacing of d_{rf} between modules. For each of these tests, Home is not considered a sensor unit, so we have a total of n sensors where $n = m^2 - 1$. In most cases, a corner node was selected to be the Home unit as depicted in Figure 2.7.

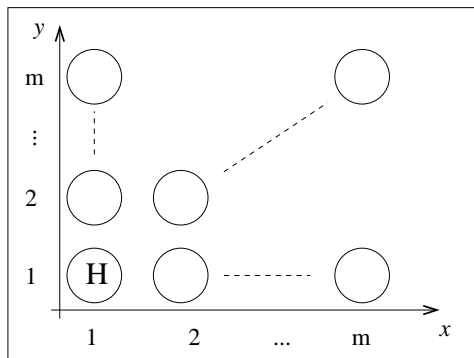


Figure 2.7: A sample network topology used for throughput testing

2.5.2 Throughput

Sensor network *throughput* is defined as the percentage of all network sensor data packets that successfully arrives error free to the Home unit. For the first throughput

test, data throughput was measured as a function of n , the number of sensor modules in the network. This series of tests used the network configuration shown in Figure 2.7 with parameter m changing for each test. In this initial topology setup, the Home unit was placed in the corner to represent a network of sensors radiating out from the Home unit in a 90° arc.

Figure 2.8 characterizes the throughput for a varying number of sensors in the network. From this plot, we see the total network throughput begins to diminish as the number of nodes in the network increases⁶. We credit this to the bottleneck effect caused by this particular network topology. Referring back to Figure 2.7 we expect bottlenecks to have a high probability of occurrence in modules (1, 2) and (2, 1). Due to their positioning, all data packets will eventually route through these two modules, in order to reach Home. Thus, modules (1, 2) and (2, 1) carry most of the workload in the network. Clearly, as the network topology increases, so does the workload placed on these two modules. The current implementation of the data queue allows for buffer overflow with overwrites. Therefore, unless the data queue is sufficiently large enough, older data packets will be dropped by these two modules, due to their higher workload. This will inevitably result in a lower total network throughput.

In an attempt to reduce bottlenecks, the next throughput test involved keeping the number of sensor modules in the network constant, but changing the position of

⁶Note, the data point for the 24 node case may not be a truly valid point. There is reason to believe complexity of the test may have resulted in performance limitation. The point does, however, give a true depiction of the behavior of the network. It is believed the true throughput value for this 24 node data point will fall slightly higher, resulting in a less dramatic drop off.

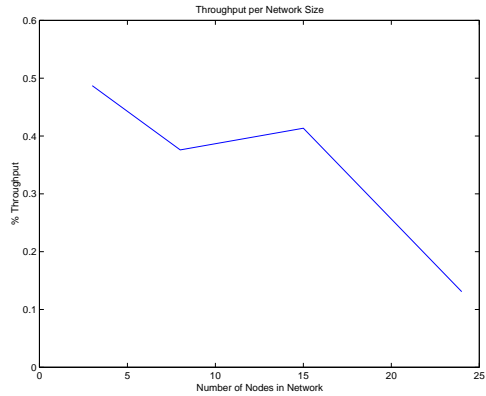


Figure 2.8: Data throughput vs. number of nodes

the Home unit. As depicted in Figure 2.9, in a grid of sensors with $m = 3$, the Home unit has three possible unique positions: the corner, the edge, and the center node.

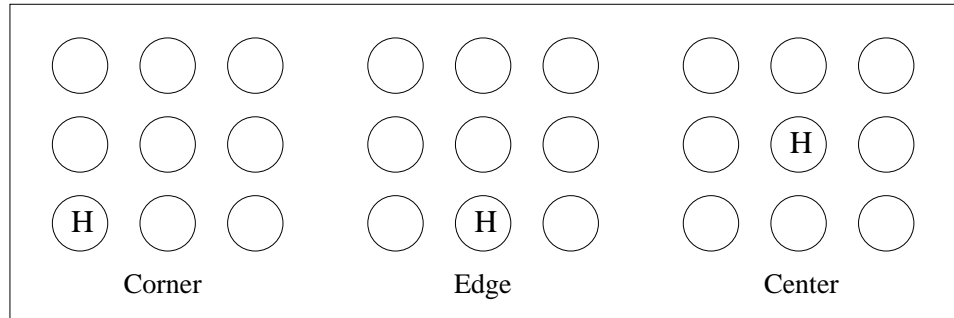


Figure 2.9: Home location testing network topologies

The results of the tests, displayed in Figure 2.10 indicates that all three topologies have comparable throughput levels. Detailed per node information of the network traffic plotted in Figure 2.11 shows the individual percentage of data throughput contribution from each module in the network topology. At a quick glance one can see that the total average throughput is almost equal for each topology.

When the Home unit is placed in the corner or center, the network throughput is generally evenly distributed. However, when the Home unit is placed on the network

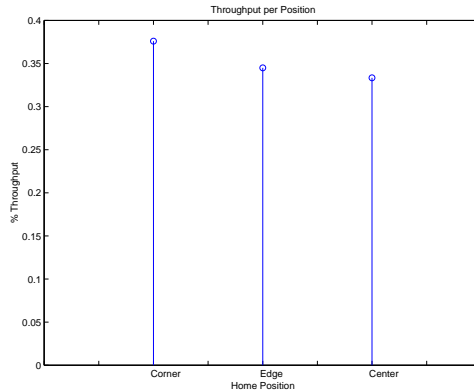


Figure 2.10: Data throughput vs. position of Home

edge as in Figure 2.11(b), the center module’s throughput is significantly less than all other modules. We credit this to the fact that the center module can communicate with all other modules, greatly increasing the probability of packets routing through the center node. This large bottleneck results in a work overload for this module. Interestingly enough, despite this large bottleneck, the total network throughput is still comparable to the other topologies, suggesting the network was able to adapt to this impedance in order to maintain total throughput. While data throughput analysis shows the three topologies to be equivalent, further investigation is required to validate this claim.

The last test was to compare the data throughput as the size of the modules’ data queue changed. When a sensor module creates or receives a data packet, the packet is placed in a local data queue. Once the module enters Client state and establishes a lock with a valid Server, the Client module transmits a random number of data packets from this data queue. The behavior of the queue is a First-In First-Out (FIFO) service with allowed overwrites. Specifically, once the queue completely fills

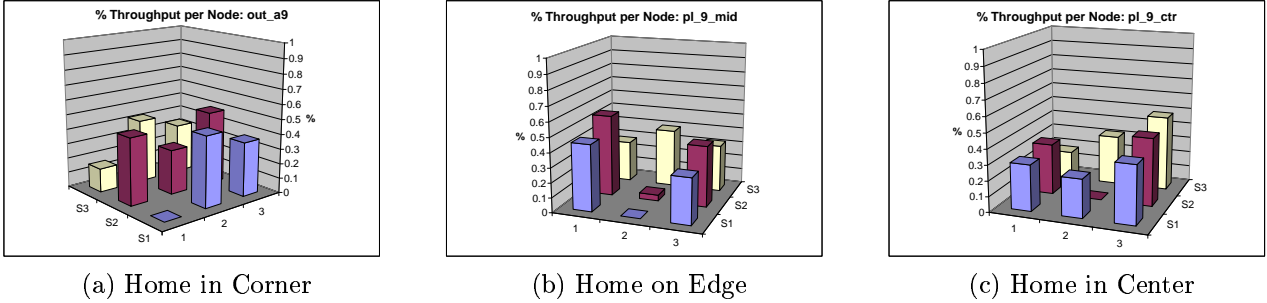


Figure 2.11: Percentage of data throughput per module

up, and new data packets are inserted into the queue, older packets are overwritten due to the queue looping. Thus, the packets that were in the queue for the longest duration are the first to be overwritten.

Figure 2.12 validates the fact that the throughput is affected by the queue size. In this test, each module generated a fixed number of data packets, while the queue size was modified. For smaller queue sizes, data packets are more likely to be overwritten and thus dropped from the network traffic. Note that the plot appears to indicate that an ideal queue size is at about 125% of the total generated packets. This suggests that other affects such as synchronicity between modules and packet collisions may be contributing to the loss in throughput beyond the queuing effects. Clearly, more results are required to validate these claims.

The figure suggests that the throughput will reach a maximum level while larger queue sizes may have negative effects on throughput. There is a high probability the data inflow rate will exceed the data outflow rate resulting in a build up in the queue.

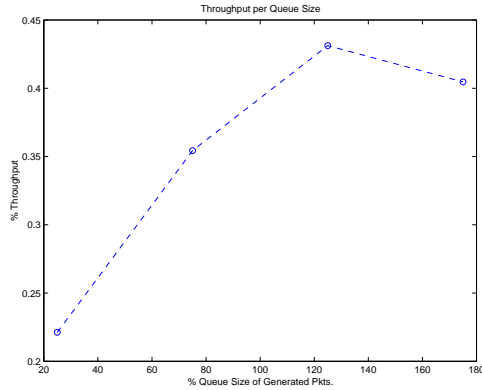
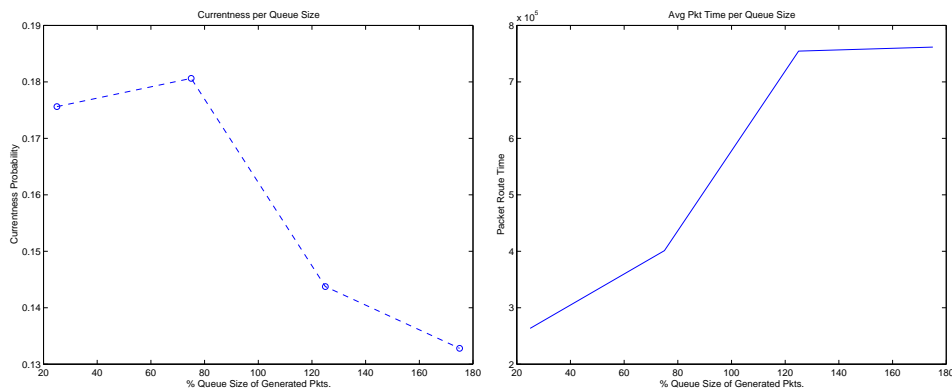


Figure 2.12: Data throughput as queue size changes

2.5.3 Currentness

As explained earlier in Section 2.4.2, the idea of currentness can be very useful for analysis of sensor data networks. For SASNs, data from the network to be received by Home does not need to be real-time. However, some metric for relevance of the sensor information in the network is needed. Specifically, we examine currentness. Figure 2.13(a) shows the results of a queue size versus currentness test. As expected, when the queue size for each module increases, the probability, α , of the received data packets being β -current reduces, where $\beta = 5$ minutes. With larger queue sizes, data packets have a higher likelihood of sitting in the queue before being served, thus increasing the total route time for the packets.

Another interesting graph is Figure 2.14, a plot of the data currentness as the position of the Home unit changes. In this test, the position of Home was changed following the three network topologies depicted in Figure 2.9. The plot in Figure 2.14 shows when Home is placed along the edge of the network, the total β -current



(a) Currentness vs. queue size

(b) Avg. route time vs. queue size

Figure 2.13: Currentness and Average Route times vs. queue size

probability of the network is greatly increased. Looking at the two topologies in Figure 2.9, with Home on the edge and in the center, we see the outer modules have roughly the same tier levels for both cases. However, in the latter case, with the Home unit in the center, all of the modules are on the network perimeter. This causes a lack of directionality in the network. It is therefore possible for data packets to circle completely around the perimeter before finally reaching Home. Meanwhile, in the former topology, when the Home unit is on the edge, there is directionality, and so packets are more likely to take a direct path towards home, resulting in a shorter lifetime of packets in the network, and ultimately, a higher β -current probability. Figures 2.15(a) and 2.15(b) validate this claim.

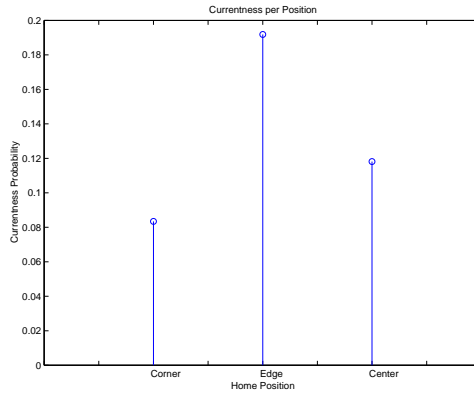
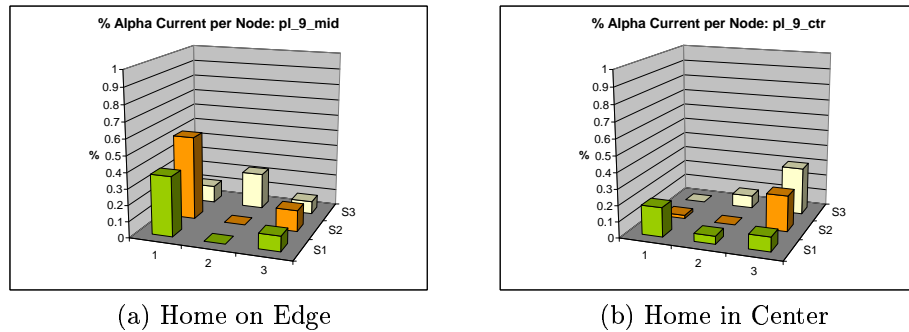


Figure 2.14: Currentness versus position



(a) Home on Edge

(b) Home in Center

Figure 2.15: Currentness per Home position

2.5.4 Average Route Time

The average route time for a module w is defined as,

$$T_{avg_p_rcvd}(w) = \frac{1}{N_{p_rcvd}(w)} \cdot \sum_{x=1}^{N_{p_rcvd}(w)} T_{p_rcvd}(x, w),$$

where $T_{p_rcvd}(x, w)$ is the time to route for packet x from module w . A graph of average route times for a network topology as the queue size for each module in the network

changes, is shown in Figure 2.13(b). Comparing this graph to Figure 2.13(a) we see that there is a direct correlation. As the average route time for a packet increases, the probability of packets being β -current decreases. This is self explanatory.

2.5.5 Load Balancing and Performance

Load balancing is especially important when using a routing protocol as dynamic as BEHGR. If two or more modules are biased and continuously synchronize exclusively with each other, an unbalanced network could form resulting in a potential overall loss in total network performance. Results from tests performed show that while nodes may bias towards one or several other nodes in specific cases, the overall statistical connection sets of neighbors are reasonably distributed as shown in Figure 2.16. This graph shows the number of local packets sent from each node. Note that the network is evenly balanced down the center, i.e. modules on the left side of the network are transmitting nearly an equal number of packets as the right side. Moreover, as the distance from Home and the tier count increases, the percentage of local packets sent also increases. Since modules at the perimeter of the network are less probable to receive packets from other modules, packets generated by an outer perimeter module have a higher likelihood of being transmitted.

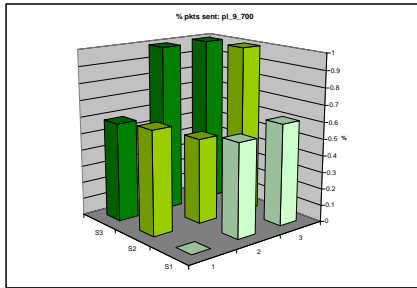


Figure 2.16: Load balancing in a network topology

2.6 Discussion

Regardless of the routing algorithm used, sensor modules closest to the Home unit will always be worked more than modules furthest away from the Home unit. There is a possibility for a bottleneck to occur at these lower tier modules, causing them to be worked substantially more than modules at higher tiers. This can lead to a shorter lifetime of the modules, as they will have more RF transmissions, resulting in more battery power consumption. To limit the occurrence of some modules having shorter lifetimes than others, an ideal network would either have the Home unit move its location, or have more Home units strategically placed throughout the network. This would shift the directionality of the network and ultimately which modules would be closest to a Home, thus evenly distributing the workload.

Due to the dynamic nature of link discovery, there is a high feasibility for packets to take alternate paths towards their destination and ultimately arrive at different times and out of order. This concept is not new and has been proven effective in the well known protocol, User Data Protocol (UDP), commonly associated with IP.

Interestingly enough, UDP is also a connectionless protocol with no guaranteed delivery.

Chapter 3

Conclusion

As a result of this research, several accomplishments were achieved. First, we presented a new paradigm for ad hoc sensor networks. By downplaying the necessity for a high quality of service, statistically accurate information can still be acquired. In a very large scale network of sensor nodes with limited capabilities the aggregated affect can be very powerful [16].

Further, some interesting results were presented regarding the use of GPS for location awareness. A sufficient probability distribution function for the time to acquire a First Lock was obtained. Location aware dependent ad hoc routing protocols can utilize this distribution for better optimizing the mobility aspect of the network. In particular careful scheduling of GPS acquisition events may be modeled to optimize power utilization while ensuring accurate location information. In addition, the TTFL distribution may be utilized for explicitly improving tracking module movement.

Finally, we introduced a new routing protocol utilizing the idea of location awareness while exploiting the nature of a SASN. Empirical testing results demonstrated viability of the protocol's use in ad hoc sensor networks. Future work is needed in refining BEHGR as a robust routing protocol for dynamic ad hoc sensor networks.

While the BEHGR algorithm fulfills the description of a Statistically Accurate Sensor Network, there are certainly other pre-existing routing algorithms which could be considered a SASN routing protocol as well. Evaluation of SASN routing algorithms for an ideal protocol is beyond the scope of this project, and would require future work.

Future development for the BEHGR routing protocol:

1. More field testing - I would like to do more extensive field testing for both larger grid array sizes, as well as repeat previous tests.
2. Simulation - For more detailed testing of large scale networks, it would be beneficial to port the BEHGR algorithm to a simulation package such as the Scalable Simulation Framework (SSF). As mentioned in the introduction, very large scale networks such as the Smart Dust project is an excellent application for SASNs and BEHGR. To prove BEHGR as a viable routing solution, very large scale routing testing needs to be performed.
3. Experiment with wider variety of COTS components - To prove the true modularity of the physical sensors, evaluation of other off the shelf components, such as other RF transceivers, is necessary.

Future research directions:

1. Mobility and Scalability - What happens when the sensor network grows to a very large size? How does BEHGR perform when the sensor modules become mobile?
2. Load Balancing and Queuing Theory - When analyzing plots of load balancing performance, very interesting behaviors occurred. I would like to do more detailed logging of each module's oscillation periods and the transactions that are performed each period. This will assist in obtaining a sense of the behavior of the network and how the network adjusts to dynamic events.

The idea of optimizing the data queue has a lot of room for further investigation. Several other queuing models could be applied and tested for performance. A closed loop form for data flow rate would help determine the ideal queue size for desired performance parameters. The tradeoff of currentness versus throughput deserves further study.

3. Power Aware Routing - We could consider an enhancement to the protocol called *BEHGR with Network Lifetime Preservation*. Since sensor modules typically have limited battery power, there has been a great deal of research in investigating the idea of power aware routing. Integrating the concept of power aware routing with the BEHGR protocol might involve each module appending the module's current power availability along with the presently transmitted

tier level k value.

Client state modules could use this added information for proper load balancing of the network, i.e. a Client would only send data packets to a Server with both a lower k value and an acceptable P value, where P is the current power availability of a Server. Qun Li investigates power aware routing to maximize the lifetime of the network [11].

Appendix A

Background Philosophy of the BEHGR Protocol

We now address some of the reasoning behind the BEHGR algorithm. Recall from Section 2.2.2, that the time duration for the Client state, $\hat{C}(j, k)$ is a direct result of the chronological history of events from previous Client state oscillation periods. When a Client does not find a lock with a Server, it is assumed the Client is oscillating at too high a frequency thus must not be waiting long enough to find a Server. For this reason, the value of T_{step} is added to $\hat{C}(j, k)$, resulting in a slower oscillation frequency.

When a Client does lock with a Server, and a valid connection is determined, this signifies a good frequency and therefore the Client module tries to maintain this synchronization by not changing any of its time duration parameters. On the

contrary, if a Client does lock, but after negotiation, the lock is determined invalid, the Client reduces its $\hat{C}(j, k)$ value by T_{step} . When this occurs, it is assumed the Client is synchronized to an incorrect tier level, and must scan for a correct synchronization by changing its current frequency¹.

¹This idea is similar to a user “scanning” a range of frequencies on a CB radio for an acceptable conversation

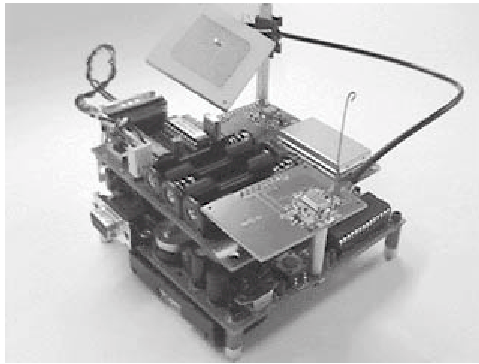
Appendix B

Sensor Module Test Bed Hardware Design

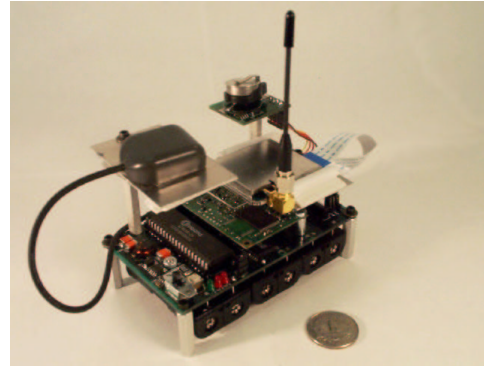
For the purpose of empirical testing, physical sensor modules were designed and built. Over the course of this research, two sensor module designs were created and are shown in Figures B.1(a) and (b). Both sensor designs are functionally identical with a hardware block diagram shown for both designs in Figure B.2. The major differences between designs are specific component selection. In the first design, all IC components are thru hole, while in the second design, all IC components are surface mount. The choice of IC components for the second design allows for a reduced physical size for each sensor module.

Another major difference between designs is the choice of RF transceiver. In the first design, a simple FM transceiver was chosen. This allowed for fast development

time. The second design uses a more complex Spread Spectrum transceiver, which allowed more robust communication among sensor modules.



(a) Version 1.0



(b) Version 2.0

Figure B.1: Two sensor designs exploiting modularity

B.0.1 Hardware Design

The physical design for the sensor modules incorporates several ideas and philosophies. First, the smart sensor modules are all identical, both in regards to hardware and software. The homogeneous design eliminates possible inherent hierarchical dependencies between sensor modules. The idea here is that any module can take the place of any other module, thus eliminating the possibility of bringing an entire sensor network down if one or few modules are “removed.”

Second, the sensor modules are modular in design. They are composed of fundamental functional blocks that are primarily commercial-off-the-shelf (COTS) components. Each unit as depicted in Figure B.2 is equipped with an off-the-shelf micro-

processor, RF transceiver, GPS receiver, and sensor. Modularity allows any one of these components to be replaced with another of the same kind (i.e. a temperature sensor may be replaced with a motion detection sensor).

The concept of integrating intelligence into the sensor (i.e. smart sensors [9]) such as a processor, memory, and other peripheral circuitry is not new, and in general, considerable research has focused on flexible ASIC design for sensors [4]. However, because each of our units is modular in design and consist of off-the-shelf components, this allows for reduced development cost and time (off-the-shelf parts are readily available and fairly inexpensive), and ease of replacement or enhancement (i.e. exchange type of RF transceiver) to meet specific mission project goals. This idea allows reusability and “reconfigurability” of the same sensors for multiple missions, resulting in lower total costs.

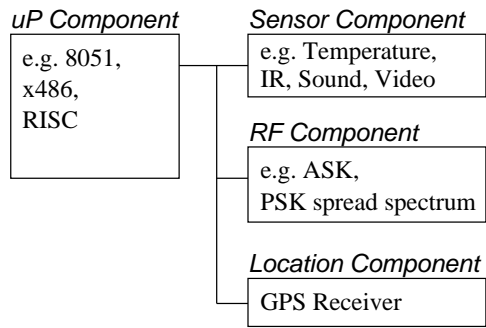


Figure B.2: A sensor design exploiting modularity

B.1 Hardware

This section will briefly describe each of the hardware components in more detail, as well as explain the necessity of each.

B.1.1 Microprocessor

Both sensor unit configurations, shown in Figures B.1(a) and (b), include an Intel 80C51 8-bit microprocessor, with internal program ROM, for controlling the sensor unit and its components. The primary responsibility of the microprocessor is to control all of the individual sub-components of the module. Very little signal processing or extensive computation is required. This micro-controller was found to be ideal because of its internal program space and multiple I/O control lines. By embedding the software code into the micro-controller, an external ROM IC component was not required, resulting in a lower chip count, smaller size, and ultimately, lower battery drain requirement. The multiple I/O control lines of the micro-controller allowed for ease of control of the separate sub-components.

B.1.2 Radio Transceiver

For communication between modules, some sort of communication device is necessary. For this type of application, where the modules may be deployed in an unknown environment, a reliable communication system is necessary. For our development, two types of radio frequency (RF) transceivers were evaluated. The first used Amplitude

Shift Key (ASK) modulation operating at a base frequency of 915Mhz. The second unit was a Spread Spectrum transceiver using Frequency Hop modulation operating in the frequency band of 902Mhz to 928Mhz. Both units had comparable transmission rates. The first unit was very effective for peer-to-peer communication, but with all units operating on the same frequency, packet collisions were frequent. Common practice of random backoff with carrier sense reduced collisions, but did not prevent them.

The second unit performed much better with regards to packet collisions and bit errors. This Spread Spectrum unit also had better performance with respect to transmission range, given the application environment. A major limitation of this second RF unit involved the embedded firmware. The current firmware restricted peer-to-peer communication to only Client-Server communication.

B.1.3 Global Positioning System

Each sensor unit is equipped with a Global Positioning System (GPS) receiver, for the purpose of obtaining position information for module position awareness. The GPS system was found to be effective for the purpose of location awareness. Again, two different GPS receivers were evaluated. The first sensor design used a receiver with a maximum channel count of 8 and an advertised mean Cold TTFF time of 90 seconds. The second sensor design used a newer GPS receiver module with the capability of tracking up to 12 channels and an advertised mean Cold TTFF time of

60 seconds. Both models proved effective and accurate. It was found through testing that placement and size of a ground plane for the GPS receiver antenna had a greater affect on performance than the choice of GPS receiver model.

B.1.4 Sensor

Both sensor module designs use a temperature sensor. Temperature sensing is performed with an 8-bit *iButton*TM sensor. In particular, this iButton technology¹, developed by Dallas Semiconductor, consists of easily replaceable and interchangeable small canisters (roughly the size of a button) with a variety of capabilities including stored memory, sensing, and network connectivity.

¹iButton Technology URL: <http://www.ibutton.com>

Appendix C

Connection Oriented Location

Aware Ad Hoc Sensor Protocol ¹

GAaRP (*Geographic Addressing and Routing Protocol*), the routing protocol used for our initial sensor network, also leverages GPS position information, but uses this information in a slightly different manner. In GAaRP, each sensor module utilizes the GPS receiver to acquire its current position at a regular interval. The unit's position, within some threshold factor, is then used as its identification, synonymous to an IP address. Sensor units utilize their newly acquired address as an identifier for routing and communication among each other. This idea differs from IP addressing in that the address is not fixed and will change as the unit's location changes.

¹Portions of this chapter have previously been published in *The Proceedings of SPIE, Enabling Technologies for Law Enforcement and Security* and presented by Michael G. Corr in November of 2000

The GAaRP routing protocol begins with each sensor unit starting in a *Module* state. Once deployed in the field, the sensor unit obtains its location, and hence its address. The unit is then ready to establish a route with a neighboring unit. A neighbor unit is defined as a sensor unit, w' , within RF propagation range of a sensor unit, w . Let $\hat{N}(w)$ be the set of all sensor units characterized as a neighbor for unit w , where $N(w)$ is the number of neighbor units in the set $\hat{N}(w)$. Let w' be a neighbor unit in the set $\hat{N}(w)$. Upon establishing this route with a neighbor, the neighbor unit will change states and becomes a *Hub*. As a Hub, a sensor unit is responsible for forwarding all data from sensor units the Hub has established routes with. A Hub may have several potential routes with other Modules and other Hubs, since all sensor units are identical, any Module may become a Hub.

The selection process for establishing the route, called the “*Who’s Out There?*” *cost function* (WOT-CF), is based on a cost function algorithm described in detail in Section C.1. This cost function algorithm is used to locate all neighbor sensor units and select the most adequate neighbor to establish a route through. However, to save superfluous RF transmissions, this location and selection process cannot commence for a sensor unit until the unit is aware of at least one neighbor unit. Thus, a sensor unit waits in Module state until it “hears” another sensor unit, through the use of a *Spark* RF message, i.e. there is at least one other sensor unit in RF propagation range.

Since all sensor Modules are initially waiting to hear this Spark message, there

must be some initial Spark from a non-Module. Thus, a Spark message from a Home terminal “ignites” the network to begin the routing process. Initially, the sensor units have no a priori knowledge of a Home unit’s position. As part of the route establishment process, the location information of the Home unit is propagated. A sensor Module cannot select the optimal route without knowing Home’s location information, therefore since all Modules are waiting for this information, the Home unit must be the one to initiate the entire routing process.

Once the route selection process is complete and a neighbor unit has been selected as a Hub for a new Module, a route is established between the two units and this new Module officially joins the network. As a new member of the network, the Module transmits a Spark RF message (identical to the one sent by Home) for other idle Modules to hear.

The algorithm repeats for each new Module as it hears the network, propagating in a radial pattern from the Home terminal outwards towards the outermost sensor unit.

C.1 “Who’s Out There?” cost function (WOT-CF) algorithm

In order for a Module unit to pick the optimal neighbor sensor unit to establish a route with, there must be some logical means for selection. In GAaRP, we use a cost

function (WOT-CF) to weigh different properties of each neighbor unit. The data values and equated results for the WOT-CF are determined through a series of RF messages and acknowledgements called “*Who’s Out There?*” (WOT), and “*Who’s Out There?*” - *Acknowledge* (WOT-ACK), respectively. The neighbor unit with the best *Cost Function Value* (CFV) is considered the optimal selection for a route according to the current network topology. As the network topology changes, so does a Module’s CFV.

To start the Hub selection process, a new Module, w , sends a WOT message containing w ’s GPS address and *Current Cost Function Value* (CCFV). The CCFV is initially set to a worst case maximum. After sending the WOT message, Module w waits and listens for the proper acknowledgments (WOT-ACK) from the neighbors, $\hat{N}(w)$.

When a neighbor unit, w' , hears a WOT message from a unit w , unit w' extracts the GPS address data of w from the WOT message and calculates a unique CFV for w' with respect to w , using the following equation:

$$CFV(w, w') = C_1 * D_{rad}(w, w') + C_2 * A_{vect}(h, w') + C_3 * N_{hops}(h, w') + C_4 * N_{slots}(w') \quad (C.1)$$

where $D_{rad}(w, w')$ is the absolute radial distance between units w and w' , $A_{vect}(h, w')$ is the delta angle vector between the neighbor unit w' and Home terminal h , $N_{hops}(h, w')$

is the number of hops a neighbor unit, w' , has to reach Home h , $N_{slots}(w')$ is the number of established routes a neighbor unit, w' , currently has allocated, and C_1, C_2, C_3 and C_4 are weighted constants calibrated for the current topology of the network.

If after evaluating (C.1), a neighbor w' calculates a better CFV than the CCFV found in the new Module w 's WOT message, neighbor w' will respond with a WOT-ACK message. Initially, all eligible neighbors will reply because the initial CCFV is set to a worst case maximum. Within the WOT-ACK reply message, a neighbor unit, w' , will include its GPS address, Home's GPS address, and the unique $CFV(w, w')$ calculated.

When the new Module w receives a WOT-ACK message from a neighbor unit, w' , Module w extracts the $CFV(w, w')$, GPS address, and Home address for unit w' from the WOT-ACK message. This information is then retained for the selection stage of the WOT-CF process. Afterwards, Module w returns to listening for other incoming WOT-ACK messages.

An important factor to note is that a neighbor w' , cannot reply to a WOT message from w , regardless of the $CFV(w, w')$ value, if w' does not already have an established route with a Hub unit or Home terminal. As a result of this policy, the first routes to be established will be between a Home terminal and a Home terminal's neighboring sensor Modules, $\hat{N}(h)$. From there, all other route links will grow in a radial pattern.

After all WOT-ACK messages have been received, the new Module w , enters the Selection stage and must now select a neighbor unit to become its Hub. Let $\hat{M}(w)$

be the set of all properly received module WOT-ACK messages by Module w , where $\hat{M}(w) \subseteq \hat{N}(w)$.

Module w sends a unicast RF message called, “*BeMyHub?*” (BMH) to the neighbor unit m with the best CFV, where $m \in \hat{M}(w)$. The neighbor, m , receiving the BMH message checks to ensure it has adequate slots still available and responds accordingly. If the answer is “No”, Module w repeats the BMH process for the neighbor unit in the set $\hat{M}(w)$ with the next best CFV. This continues until a positive response is received. A route is established after Module w sends a final “*YouAreMyHub*” message to the neighbor unit, m . If a Home terminal is the recipient of a BMH message, the Home terminal will unconditionally reply with a positive response.

Once this final handshake is complete, the new Module w is considered part of the network and must now announce its presence to other Modules still idle. To do so, Module w broadcasts the same Spark RF message sent by the Home terminal. All Modules in RF transmission range without a route will now start their own Hub selection process. In addition, Module w retains the GPS address and state information for the new route established between w and m . All other sensor unit information of neighbor units in $\hat{M}(w)$ is disregarded for the interest in saving RAM space.

C.1.1 Cost Function Example

Figure C.1 demonstrates in more detail, the purpose of each of the four parameters found in (C.1). In this figure, we see an established routing topology with a new

Module N attempting to enter the network. It is assumed N can hear Modules C, E , and G . When selecting a neighbor unit to be a Hub, one with the highest reliability is considered the best. Reliability is judged on geographic positioning and current status of the sensor unit. In more detail, the closer the distance between two units, the better the chance of an RF transmission getting through, demonstrated in Figure C.1(a). By taking further advantage of position information, a sensor Module can select a neighbor unit which is in the direction towards a Home terminal, Figure C.1(b). This does not guarantee an optimal route, but helps. The third parameter measures unit hops to a Home terminal. The fewer number of hops an RF message has to take, the less susceptible an RF transmission is to corruption or packet loss. Also, when choosing a neighbor unit to be a Hub, the algorithm will favor a unit with a lower workload status by selecting a Hub with fewer established routes as in Figure C.1(c). This attempts to address the issues of fairness and load balancing.

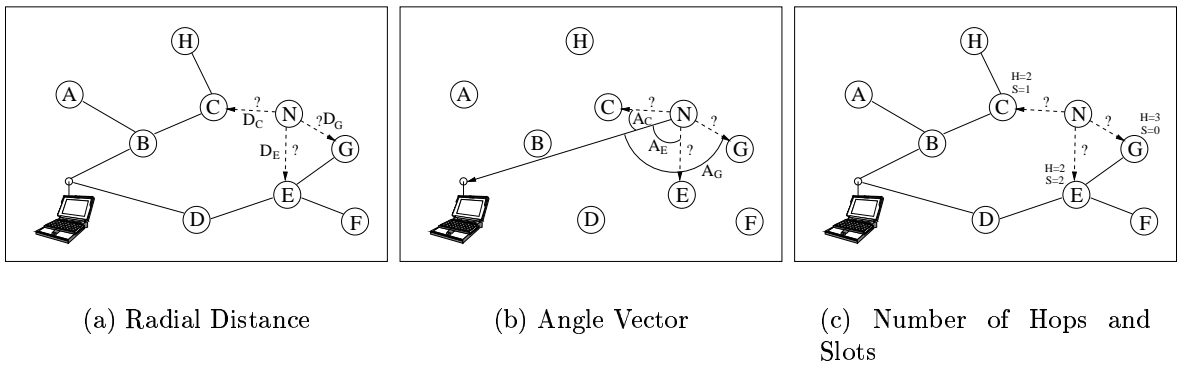


Figure C.1: Cost Function Value Algorithm

One can easily envision several network topology scenarios where each of these four parameters has a higher importance than the others. For this reason, we use

coefficients to dynamically weight the different parameters depending on what type of environment the sensor units will be deployed in.

Example 4 (GAaRP Example) Consider an example of GAaRP. In the first frame, Fig. C.2(a), we see several Modules sitting idle, while three Modules, A, B, and C are sending an initial WOT message to initiate the Hub selection process. It is assumed the Home terminal (laptop) has already sent a Spark message and only modules A, B, and C where able to hear it.

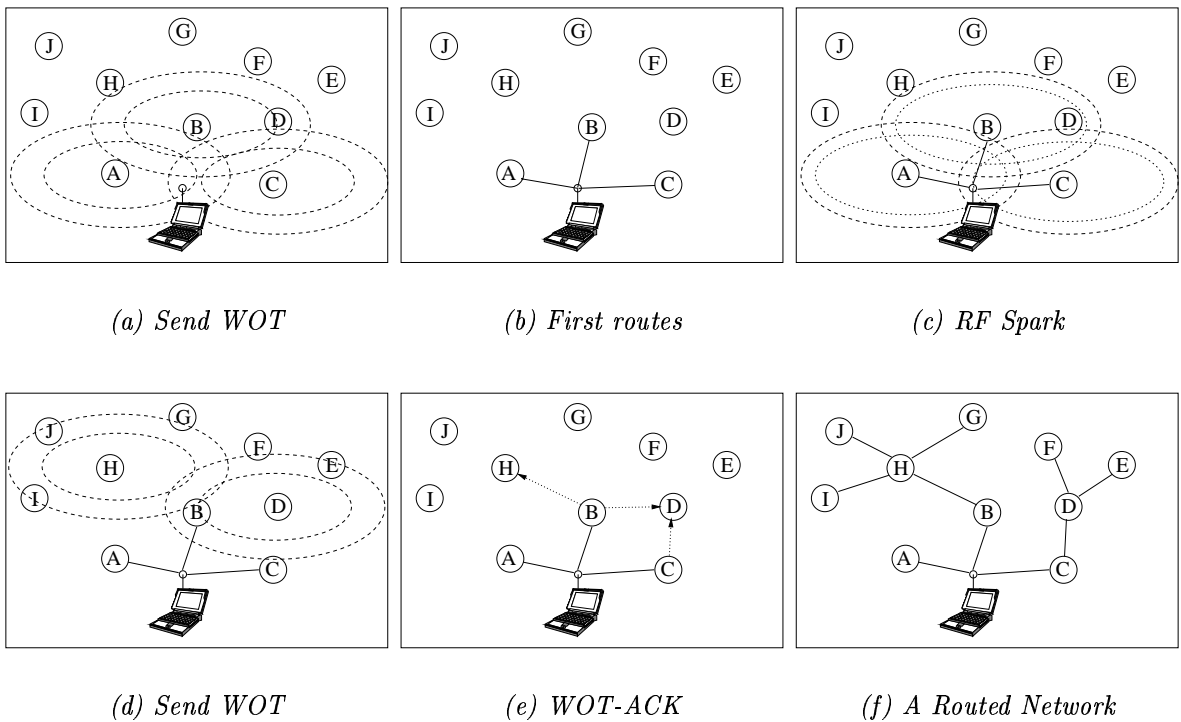


Figure C.2: Geographic Address and Routing Protocol

Since this is the first route to be set up, only Home will be able to reply with a WOT-ACK message. Modules D and H can both hear the WOT messages, but have no established routes themselves, so cannot reply. Thus, Modules A, B, and C have

no choice but to set up a route with the Home terminal, as depicted in Fig. C.2(b). After these routes are set up, the three Modules are considered part of the Network and now send out their own Spark message, Fig. C.2(c).

Modules *D* and *H* hear the new Spark messages, so they send out their own WOT message, Fig. C.2(d). Two eligible Modules, *B* and *C*, hear the WOT message from Module *D*, so they both calculate a CFV and reply with a WOT-ACK to *D*, Fig. C.2(e). Meanwhile, *B* can also hear a WOT message from *H*, so *B* replies with a unique WOT-ACK message and CFV to *H*, as well.

We see in Figure C.2(f) that Module *D* picked *C* to be its Hub. This figure also shows what a final routing scheme might look like for this sensor network topology, after the Hub selection process was evaluated by all remaining sensors units.

C.2 Simulation Results

To evaluate the performance of GAaRP, the protocol was ported to a simulation package, Ad Hoc Network Simulation Package (AHNSP)¹. The simulation package was responsible for randomly creating a network topology and simulating the GAaRP route establishment process, using given network conditions entered into a configuration file. This configuration file allows the user to specify the number of nodes, the 2-dimensional area, the number of simulation rounds, and the RF transceiver propagation distance for the wireless sensor modules.

¹AHNSP was developed by Jeremy Robin of Dartmouth College

C.2.1 Route Link Setup

Simulation results for average time to route per node are plotted in Figure C.3. The time to route includes the duration from when a Module initially receives a *Spark* RF message until the “*YouAreMyHub*” message is sent. This includes all negotiations and restarts due to timeouts and packet loss. The x axis of the graph shows the total number of nodes in the network topology, with the y axis showing the average route time for each node in the corresponding topology. As previously expected, and proven in the graph: when the number of nodes increases in a fixed area, the average route time per node will increase. In other words, as the density of nodes increases, the probability of packets colliding also increases. Packet collisions lead to packet loss. This loss of packets means nodes may not receive awaited messages within an allotted time, resulting in timeouts and restarts and ultimately, a longer route establishment process.

Fortunately, Figure C.3 is linear. As the number of nodes in the network increases, the corresponding route times increases linearly. We hypothesized that the route times will reach a maximum and level off. Had the curve been of exponential nature, this would be less likely to occur.

C.2.2 Routability and Network Density

An important aspect of sensor networks is the *routability* factor. In a static network, the density of sensor modules must be large enough for all units to have access to a

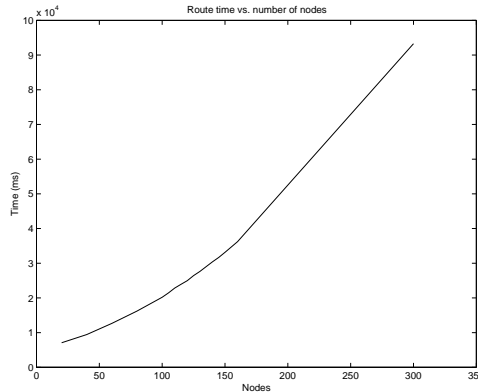


Figure C.3: Average time to route per node

route. In a mobile network it is permissible for this density level to be lower, assuming modules will be moving into position allowing route links to eventually be established.

Regardless of the routing protocol used, if a sensor module, w , is out of RF propagation range from all other modules or access points, module w will be deemed *radio isolated* and thus unroutable.

For simulation purposes, a fixed square network region of 200 yards by 200 yards was chosen, with a sensor module RF propagation range of 30 yards. Figure C.4 shows the plots of what percentage of nodes were routable as the module density increased. The plot also shows what percentage of nodes actually did route.

This plot shows promising results. When the number of randomly distributed nodes in the network area is small, there is a higher probability for several nodes to be radio isolated, thus resulting in a lower network routability percentage. Fortunately, the plot shows this percentage value increases rapidly as the network density increases. This value levels off at about 150 nodes in a $40000yd.^2$ area, or a node density of 0.00375 nodes per square yard. This is good, as it shows the GAaRP protocol does

not need a high module density in order to get a high routability percentage value.

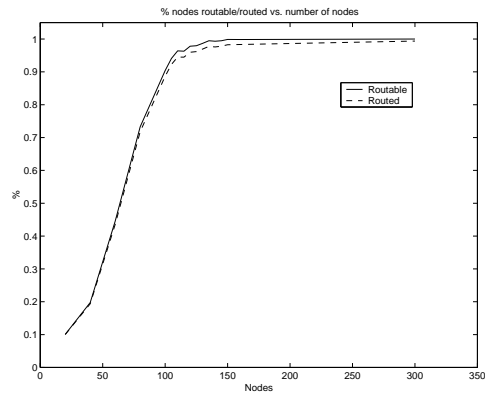


Figure C.4: Percent of nodes routable

Acknowledgements

First off, I would like to thank my advisor, Clayton Okino, for all the help and hard work he provided me throughout my residency. Without Clay, I would not be where I am today. For that I thank you.

My other committee members, George Cybenko and Bob Gray, I thank you for your help and guidance throughout the project.

Shyam Yadati, my right hand man, thank you for your willingness to help. Without you, I would not have been able to complete all the things I wanted to.

I was most impressed with you, Jeremy Robin, for making the executive decision to write your own simulation package from scratch. Remember, you helped keep GAaRP alive.

Then of course, there is Dan Burroughs, who was always willing to help at a moments notice. Thank you for your good humor and necessary sarcastic comments.

Finally, there is the rest of the gang at ISTS. I thank you all for your patience and assistance with the field testing. Most importantly, thank you everybody who helped with the assembly of the sensor modules. You all out did yourselves, proving to be excellent hardware assembly engineers.

Bibliography

- [1] K.N. Amouris, S. Papavassiliou, and Li Miao. A position-based multi-zone routing protocol for wide area mobile ad-hoc networks. In *IEEE Vehicular Technology Conference 1999*, volume 2, pages 1365–1369, 1999.
- [2] S. Basagni, I. Chlamtac, and V. R. Syrotiuk. Geographic messaging in wireless ad hoc networks. In *Proceedings of the IEEE 49th Annual International Vehicular Technology Conference (VTC'99), Houston, TX, May 16-20, 1999*, volume 3, pages 1957–1961, 1999.
- [3] S. Basagni, I. Chlamtac, V.R. Syrotiuk, and B.A. Woodward. A distance routing effect algorithm for mobility (dream). In *ACM/IEEE Int. Conf. on Mobile Computing and Networking*, pages 76–84, October 1998.
- [4] M.E.C. Bowen and G.B. Smith. Design of flexible asics including embedded processors for smart sensor applications. In *IEEE Application Specific Integrated Circuits for Measurement Systems*, pages 5/1 – 5/3, 1994.
- [5] Brian Brewington. *Observation of changing information sources*. Phd thesis, Dartmouth College, June 2000.
- [6] Michael G. Corr and C. Okino. Networking reconfigurable smart sensors. In *Proceedings of SPIE, Enabling Technologies for Law Enforcement and Security*, November 2000.
- [7] Piyush Gupta and P. R. Kumar. *Critical Power for Asymptotic Connectivity in Wireless Networks*, pages 547–566. Birkhauser, Boston, 1998.
- [8] Sam Halabi. *Internet Routing Architectures*. Cisco Press, 2nd edition, 2000.
- [9] E. Jacobsen. The building blocks of a smart sensor for distributed control networks. In *Northcon 1996*, 1996.
- [10] Y.-B. Ko and N.H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. In *ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, November 1998.

- [11] Qun Li, J. Aslam, and D. Rus. Online power-aware routing in wireless ad-hoc networks. In *To be presented at MobiComm 2001*, 2001.
- [12] R. Morris and B. Karp. Greedy perimeter state routing (gpsr). *Draft*, <http://www.eecs.harvard.edu/karp/gpsr4-98.ps>, April 1998.
- [13] J.C. Navas and T. Imielinski. Geocast - geographic addressing and routing. In *ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom'97)*, September 1997.
- [14] C. Perkins. *Ad-Hoc Networking*. Addison-Wesley, 2001.
- [15] Charles E. Perkins and Elizabeth Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, February 1999*, pages 90–100, 1999.
- [16] Gregory J. Pottie. Wireless sensor networks. In *Information Theory Workshop*, pages 139–140, 1998.
- [17] Elizabeth Royer and C-K Toh. A review of current routing protocols for ad-hoc mobile wireless networks. In *IEEE Personal Communications Magazine*, April 1999.
- [18] B. Warneke, M. Last, B. Liebowitz, and K.S.J. Pister. Smart dust: communicating with a cubic- millimeter computer. In *Computer*, volume 34, January 2001.