Networking Reconfigurable Smart Sensors†

Michael G. Corr and C. M. Okino
(*mgcorr@dartmouth.edu*) (*cokino@dartmouth.edu*)
Thayer School of Engineering
Dartmouth College
Hanover, NH 03755

ABSTRACT

The advances in sensing devices and integrated circuit technology have allowed for the development of easily "reconfigurable smart sensor" products. Primarily utilizing commercial off-the-shelf (COTS) components, we have developed reconfigurable smart sensors, consisting of a microprocessor, GPS receiver, RF transceiver, and a sensor. The standard serial control interface allows for ease of interchangeability for upgrades in RF transmission schemes as well as customizing the sensing device (i.e. temperature, video images, IR, motion, Ethernet) per application. The result is a flexible module capable of gathering sensor data, local processing, and forwarding compressed information to a central location via other modules.

In this paper, we present our system infrastructure design and a cost function based geographical self-routing algorithm for networking reconfigurable smart sensors. The algorithm allows for the sensors to automatically negotiate in a geographical radial topology relative to a central location, utilizing other sensors as routes or hops for forwarding information to a central location. A number of these sensors are currently deployed out in the field and we present performance measurements on routing and transmission of sensor data. Scalability issues are addressed in analysis of a very large scale reconfigurable smart sensor network.

Keywords: Geographical Routing, Smart Sensors, Distributed Sensing, Ad-Hoc Networks, Self Routing, Data Acquisition Networks

1. INTRODUCTION

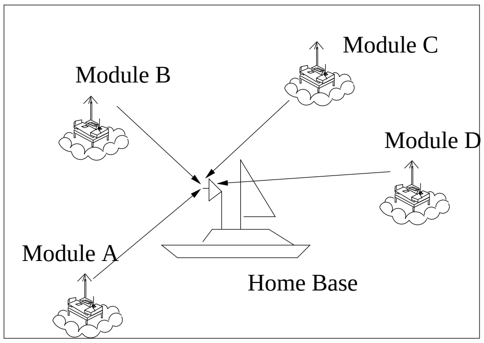
In this paper we present a smart sensor network that is based on a geographical topology of routing and networked access. Specifically, we introduce a novel approach of addressing each module purely by the module's GPS acquired position. Although utilizing the unit position information as an address location is unique, the concept of geographical routing is similar to work by Morris and Karp[8] and is in the spirit of the work in.[1] In this paper, we propose an ad-hoc self-routing algorithm called, *Geographical Addressing and Routing Protocol* (GAaRP) and present results from implementation of the algorithm on a live test-bed of smart sensors modules. We begin in section 2 by describing the design characteristics of our sensor unit and attributes found in a network topology for our application. This is followed by a detailed description, with examples, of our proposed routing scheme, in section 3.

We focus on a distributed data acquisition system allowing for consolidation of data from a safe distance e.g. brush fires, hazardous chemical spills, violent storms. The concept of independent and distributed collection and processing of sensor data in a geographically distributed topology is defined as a *Distributed*

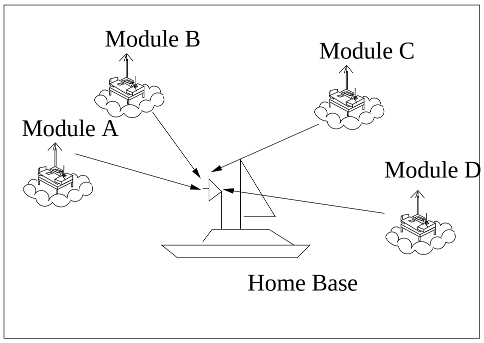
† This research was partially supported by National Science Foundation grant CCR-98113744, Air Force Office of Scientific Research Multidisciplinary University Research Initiative Grant F49620-97-1-0382,and DARPA Contract F30602-98-2-0107.

Smart Sensor Network (DSSN). Related work on the topic of very large scale networks is addressed in[4][9][5] in terms of optimizing the capacity of these wireless networks.

Consider the scenario where a number of sensors are scattered in the ocean to gather localized temperature readings. Due to uncontrollable environmental affects, the sensor network topology may be volatile. Therefore, information gathered by the sensors should be location dependent, not identity dependant as depicted in Figure 1 for sensor A drifting to a new location.



(a) initial state of sensors



(b) after sensors have drifted

Figure 1: Geographical addressing networked sensors - location oriented data acquisition

Specifically, our network topology uses location oriented identification, routing, and processing as opposed to the traditional approach of contacting and receiving data based on a source's static identification (i.e. an IP address).

2. THE HARDWARE RECONFIGURABLE SENSOR MODULE & ARCHITECTURE

The smart sensor modules are all identical, both in regards to hardware and software. The homogenous design eliminates possible inherent heirarchial dependencies between sensor modules. These sensor modules are composed of fundamental functional blocks that are primarily commercial-off-the-shelf (COTS) parts. Each unit as shown in Figure 2 is equipped with an off-the-shelf microprocessor, RF transceiver, GPS receiver, and sensor. The concept of integrating intelligence into the sensor (i.e. smart sensors[6]) such as a processor, memory, and other peripheral circuitry is not new, and in general, considerable research has focused on flexible ASIC sensors.[3] However, because each of our units is modular in design and consist of off-the-shelf components, it 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 sensor) to meet specific mission project goals (i.e. replace RF with low power spread spectrum components to minimize the effect of RF jamming and detection). This idea allows reusability of the same sensors for multiple missions, resulting in lower total costs.

Due to the simplicity of common temperature sensors, we will initially consider a model involving the monitoring of temperature data. Typically, sensors of this sort will not be used for security purposes. However, one could visualize the scenario of utilizing this type of sensor to detect heat radiating off equipment or personnel from intruders entering a restricted area. In general, the modularity of our design

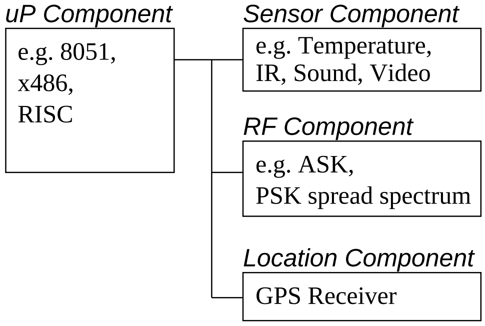


Figure 2: A sensor design exploiting modularity

allows for flexibility in varying the type of sensor in our research and development. This implies that the exact type of sensor is not critical to the nature of the study.

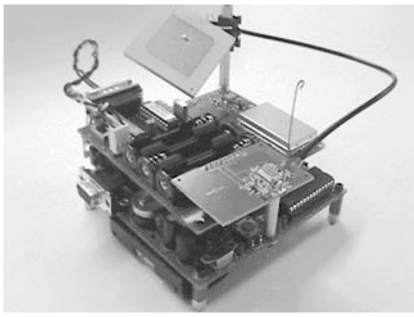


Figure 3: A typical module

The current sensor unit configuration, shown in Figure 3, includes an Intel 80C51 8-bit microprocessor, with internal program ROM, for controlling the sensor unit and its components. For communication between units and Home, we use an RF transceiver operating at a base frequency of 915Mhz, with an Amplitude Shift Keying (ASK) modulation scheme. A GPS receiver is used for location determination and addressing, as described below in Section 3.1. Temperature sensing is performed with an 8-bit *iButton*(TM) sensor. In particular, this iButton technology[*], 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.

2.1. System Level Infrastructure

To develop a distributed smart sensor network (DSSN), we propose the design of small, inexpensive, wireless units. Each unit is responsible for acquiring sensor data and then transmitting some form of the collected data through an RF wireless ad-hoc network.

After deployment into the network, each unit selects one of two states, *Module* or *Hub*. The current topology of the network will determine which state a unit is to select, while still allowing units to change state indefinitely on a periodic basis. The state of a unit determines its primary tasks and capabilities.

The main objective of each unit, regardless of state, is to collect, process, and transmit data to a local *Home* or *Gateway* terminal. A Home is defined as a location where a user is allowed to interact with the

[*]iButton Technology URL: http://www.ibutton.com

sensor network. (i.e. a laptop or PC terminal connecting to the DSSN.) A Gateway is simply an access node connecting the DSSN to an accessible LAN or possibly the Internet.

Due to the ad-hoc nature of the DSSN, some form of transmission packet formatting and processing is required. An organized scheme for self routing must be invoked, in order for the sensors to relay their collected data in an organized fasion. The following section describes a typical sensor network application and a corresponding routing scheme.

3. GEOGRAPHICAL ADDRESSING & SELF-ROUTING NETWORK

Each sensor unit utilizes a GPS receiver to acquire its current position at a regular interval. The unit's position, within some threshold factor, is used as the 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, it will change as the unit's location changes.

To reduce network congestion and allow for efficient throughput, a proficient routing protocol must be developed. The development of this algorithm takes into account the inherent characteristics of a typical sensor unit.

1. The units have limited power, as they are run off batteries. This means the algorithm must limit RF transmissions and unnecessary use of power.
2. The sensor units have limited memory (RAM) space, so cannot retain large amounts of superfluous data.
3. RF transmission ranges are limited, so it is assumed RF transmissions may have to hop through several units before reaching an intended receiver.

The routing protocol also takes into account the unique characteristics of a typical network in which the sensor units will be deployed.

1. The network is used for data acquisition, *not* for communication. Therefore, little or no communication is performed *Module-Module*. In general, all communication is either *Module-Hub, Hub-Hub,* or *Hub-Home.*
2. There are many small Sensor Units with only *one* or *few* central Home terminals.
3. When collecting data from the network, a user cares only *where* the data comes from, not *who* it comes from.
4. All sensor data is localized.(i.e. all data collection is performed at each unit, allowing each unit to require no knowledge of data collected by neighbor units.)
5. The network is polarized. (i.e. *data communication* is one way (from unit to Home) while *control communication* flows the other way, from Home to unit.

The following section presents the resulting routing protocol developed from the above mentioned network and sensor unit characteristics.

3.1. Routing Algorithm

In this section we present GAaRP *(Geographic Addressing and Routing Protocol)*, the routing protocol used for our sensor network. The routing process begins with each sensor unit starting in a *Module* state. This means the unit has no responsibilities save collecting data and listening for RF messages in the network. Once deployed into the field, a Module must find its location, and hence its address, using a GPS receiver.

After this is obtained, the goal for each Module is to find a *Link Route* with a neighbor Module. Upon establishing this link, the neighbor Module changes state and becomes a *Hub*. As a Hub, a sensor unit is responsible for forwarding all data from Modules it has established Link Routes with. A Hub may have several Link Routes with other Modules and other Hubs. All Modules are identical, so any Module may become a Hub.

The selection process for establishing the Link Route, called *WOT-ACK*, is based on a Cost Function Algorithm and is described in more detail below in section, 3.1.1. The Cost Function Algorithm is not executed until a Module can hear the *Sensor Network*, i.e. it is sure there is at least one other sensor unit in RF listening range. Until so, it sits idle waiting to hear an RF message from a neighbor Module or Hub.

If all Modules in a sensor network are sitting idle waiting to hear a neighbor's RF transmission, they will all remain idle. Thus, there needs to be some spark to start the Link Route selection process. This *Spark* comes from a *Home* terminal. When the sensor Modules are deployed, there is no a priori knowledge of a Home terminal's position. The sensor Modules cannot possibly create an efficient network routing topology until this information is known. This is why the initial RF message must come from a Home unit. It is possible for a sensor network to have multiple Home terminals, so a sensor Module will use the Cost Function Algorithm to pick the best one for forwarding data to.

Upon receiving a Spark RF message, a sensor Module can now start the Hub selection process as it can now "hear" the network. Once this process is complete and a neighbor Module is selected as a Hub by a new Module, a Route Link is established between the two and the new Module can join the network. Now it must announce it's presence to the network by sending its own RF Spark message for other Modules to hear. This allows other Module units, currently sitting idle, to begin their own Hub selection process.

The algorithm repeats for each new Module as it hears the network, propogating from the Home terminal outwards towards the outermost sensor unit.

3.1.1. Who's Out There? (WOT) Cost Function Algorithm

In order to pick the optimal neighbor Module to become a Hub, there must be some logical means for selection. In this routing protocol, we use a cost function to weigh different properties of each neighbor Module. The data values and equated results for the Cost Function are determined through a series of RF messages and acknowledgements called *Who's Out There? (WOT)*,and *WOT-ACK*, respectively. The neighbor Module with the best *Cost Function Value (CFV)* is considered the optimal selection for a Link Route according to the current network topology. As the network topology changes, a Module's CFV will update as necessary.

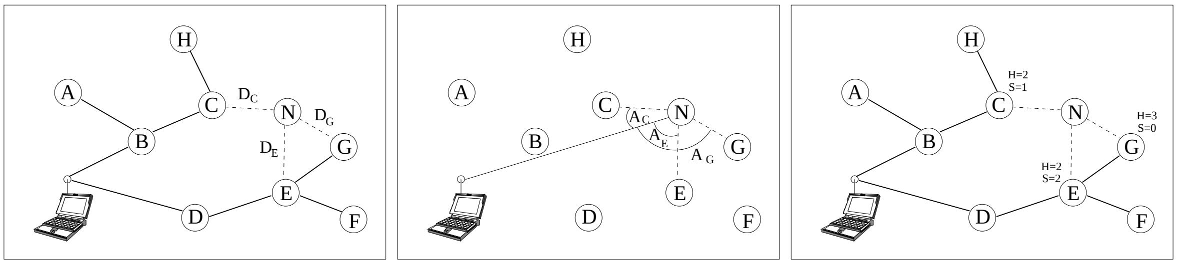
To start the Hub selection process, a new Module sends a WOT message containing its GPS address and its *Current* Cost Function Value (CCFV). The CCFV is initially set to a worst case maximum. After sending the WOT message, the Module waits and listens for the proper acknowledgments (WOT-ACK) from its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

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

$$CFV \quad = \quad C_1 * D_{radial} + C_2 * A_{vector} + C_3 * N_{hops} + C_4 * N_{slots} \tag{1}$$

where D_{radial} is the absolute radial distance between units, A_{vector} is the delta angle vector between the neighbor unit and Home terminal, N_{hops} is the number of hops neighbor unit has to reach Home, N_{slots} is the number of slots neighbor unit has allocated , and C_1, C_2, C_3 and C_4 are weighted constants calibrated for the topology of the network.

Figure 4 demonstrates the purpose of each of the four parameters found in equation (1). In this figure, we see an established routing topology with a new Module (N) trying to enter the Network. It is assumed (N) can hear Modules (C),(E), and (G). When selecting a neighbor Module 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 Fig. 4(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, Fig. 4(b). This does not guarentee an optimal route, but helps. The third parameter measures unit hops to a Home terminal. The fewer number of hops a RF message has to take, the less succeptible an RF transmission is to corruption or packet loss. Also, when choosing a neighbor Module to be a Hub, it is best to select one with fewer Link Routes already established. This concept is called *fairness*. It is better to equally distribute Link Routes throughout the network, so no single Module is overburdened, Fig. 4(c).



(a) Radial Distance (b) Angle Vector (c) Number of Hops and Slots

Figure 4: Cost Function Value Algorithm

One can easily invision 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 sensors will be deployed in.

If after evaluating equation (1), a neighbor calculates a better CFV than the CCFV found in the new Module's WOT message, the neighbor responds with a WOT-ACK message. Initially, all elligible neighbors will reply because the initial CCFV is set to a worst case maximum. Within the WOT-ACK reply message, the neighbor Module includes its GPS address, its Home's GPS address, the unique CFV it calculated, and the necessary raw data for recalculating the CFV.

An important factor to note is that a neighbor cannot reply to a WOT message, regardless of its CFV, if it does not already have an established Link Route with a Hub unit or Home terminal, itself. Without an established route towards a Home terminal, a unit is not considered part of the network, and thus does not have permission to route another Module into the network. 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. From there, all other route links can grow, as shown later in Section 3.1.2 and Figures 5(a)-(f).

When the new Module receives a WOT-ACK message, it extracts the included raw data from the message and recalculates the CFV for the corresponding neighbor unit. If the recalculated CFV matches the CFV found in the WOT-ACK message, then it is assumed the RF message was not corrupted during transmission and the CFV is valid. The new Module then records the neighbor's CFV, GPS address, and Home address, and returns to listening for other incoming WOT-ACK messages. In the case where the two CFVs do not match, the entire WOT-ACK message is disgarded.

The new Module must make sure it has acquired all WOT-ACK messages from neighbors with potential route links. To do so it sends another WOT message, after a timeout from when the last WOT-ACK message is received. This subsequent WOT message is identical to the first, except for a possibly new CCFV. The CCFV is updated with the best CFV received from all previous WOT-ACK messages received. The updating of the CCFV solves two problems. First, it helps reduce extra RF transmissions from neighbor units. If a neighbor cannot beat the CCFV, then it is not a top choice for a Route Link and therefore has no reason to reply. Second, it helps resolve RF packet collision occurrences. When a neighbor responds with a WOT-ACK message it has no way of determining if the new Module successfully received the message. If the neighbor unit sees the CCFV in a subsequent WOT message is worse than the CFV it sent in a previous WOT-ACK message, then it can assume its previous WOT-ACK message was not received, and therefore must be resent.

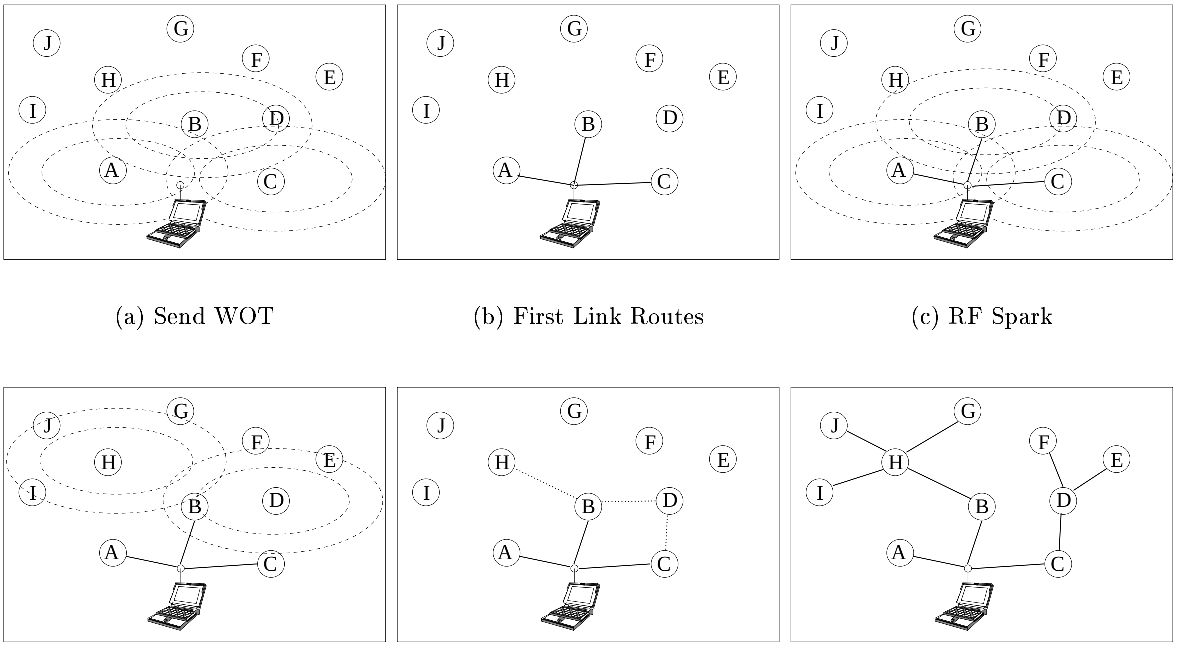
Testing thus far has determined that five rounds of WOT messages, with an adequate wait interval between rounds, is sufficient for receiving all possible WOT-ACK messages. To further reduce unnecessary RF transmissions, if no response is received in two successive rounds, the sending Module assumes there are no more WOT-ACK messages to receive, and thus quits sending WOT messages. If it has not received any WOT-ACK messages, the sensor Module returns to its idle state, waiting to hear another neighbor's RF message, to start the process again.

After all WOT-ACK messages have been received, the new Module must now select a neighbor unit to become its Hub, and thus establish a Route Link. The new Module sends a unicast RF message called, *BeMyHub? (BMH)* to the neighbor unit with the best CFV. The neighbor receiving the BMH message checks to make sure it has adequate slots still available and responds accordingly. If the answer is No, the new Module repeats the BMH process for the neighbor unit with the next best CFV. This continues until a positive response is received. The Route Link is established after the new Module sends a final *YouAreMyHub* message to the neighbor unit. If a Home terminal is the recipient of a BMH message, it will unconditionally reply with a positive response.

To save precious RAM space in the sensor unit, it will erase the entire list of WOT-ACK messages it has collected. It is not necessary to retain this information as the network may have changed drastically in the case a unit has to find a new Link Route sometime in the future. The only information a unit needs to retain about its neighbors is the information pertaining to its Hub unit, as this is the only neighbor the sensor unit corresponds with. Hence, no routing tables are required as is customary in several popular ad-hoc routing schemes.

Once this clean-up process is complete, the new Module is considered part of the network, and must now announce its presence to other Modules still sitting idle. To do so, it broadcasts the same RF Spark message sent by the Home terminal. All Modules in RF transmission range without a Route Link will now start their own Hub selection process. The new Module may now establish Link Routes through itself by becoming a Hub for other sensor Modules in the network, as necessary.

3.1.2. GAaRP Example



(a) Send WOT (b) First Link Routes (c) RF Spark

(d) Send WOT (e) WOT-ACK (f) A Routed Network

Figure 5: Geographic Address and Routing Protocol

In Figure 5 we demonstrate a simple example of the GAaRP routing protocol. In the first frame, Fig. 5(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.

Since this is the first Link 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 Link Routes themselves, so cannot reply. Thus, Modules (A), (B), and (C) have no choice but to set up a Link Route with the Home terminal, as depicted in Fig. 5(b). After these Link Routes are set up, the three Modules are considered part of the Network and now send out their own Spark message, Fig. 5(c).

Modules (D) and (H) hear the new Spark messages, so they send out their own WOT message, Fig. 5(d). Two elligible 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. 5(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 5(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.

3.1.3. Discussion on the Proposed Protocol

Allowing all units the ability to route packets results in *any* unit in the network becoming a viable link *(Hub)* to a Home. This idea has received scrutiny in the past, but unlike previous routing methods such as *distance vector* or *link state routing*,[7] our scheme does not require storage of large routing tables. Since we assume the strong condition that all information transfer is unidirectional, each Module only needs to know its corresponding Hub, eliminating the need for large routing tables.

The GAaRP protocol described above allows for packet collisions to occur without creating false connections by insuring a valid connection between a querying unit and a potential route unit. Integrating the classic collision detection schemes with random back-off, typical in ethernet and Aloha,[2] will reduce the effect of collisions and, along with CFVs, reduce the number of competing potential route units. Specific to the CFV is the nested "angle" vector (i.e. variable A_{vector} in (1)) signifying direction towards Home, eliminating many potential route units that do not hold to the directed graph assumption.

Another nice feature of the routing protocol is the non-necessity to retain large amounts of data about the network (e.g. routing tables). Only routing information for one other unit (Hub) is necessary to retain. This Hub is used for forwarding collected data to a Home terminal. In the case the Link Route to a Hub is lost (i.e. change in environment or network topology), a Module simply re-evaluates the Hub selection algorithm.

As noted above, the resulting network topology is polarized, all data packets flow from the sensor units to a Home, while all control packets flow from a Home to the specified sensor unit(s). In the latter case, control packets are not sent to an individual sensor unit, but rather a specific geographic region. For this reason, a Hub does not need to retain routing information about its Link Routes to outward sensor units. If a Hub receives a control packet forwarded from a Home terminal, it simply checks the destination region of the packet, and broadcasts the message accordingly.

4. ANALYSIS AND RESULTS

In this section we model our DSSN, perform some straightforward analysis and provide results from our actual test-bed of distributed smart sensors.

4.1. The Model

We now analyze our protocol generalizing the model in the previous section.

We recognize that the routing *setup time* for the sensor network is a function of the number of units and the topology chosen. We consider the single hop case and evaluate the time needed to establish a route for n sensor modules.

Clearly, the *setup time* assuming all units are powered on simultaneously and deployed instantaneously equates the the time required to acquire a valid GPS measurement and the required to acquire a valid route HOME.

Let T_{GPSon} be the amount of time for a unit to acquire a valid GPS location measurement and T_{route} be the amount of time required to establish a route HOME given n modules. Thus we desire to obtain

$$T_{setup} = T_{GPS} + T_{route}(n)$$

However, since each of these values are random variables, we obtain the first order statistic of their expected value, i.e.

$$E[T_{setup}] \quad = \quad E[T_{GPS}] + E[T_{route}(n)]$$

It was determined, for our specific sensor unit hardware design using a GPS receiver with a proper ground plane and antenna, that the GPS acquisition times behaved similar to a *Weibull* random variable with parameter $\alpha = 1$ and parameter $\beta = 2$. We assume a discrete time slot model where ΔT_{GPS} is the minimum incremental measurement of time. We then obtain the average number of ΔT_{GPS} time slots to establish a valid GPS address.

THEOREM 1. *The average amount of time to acquire a valid GPS measurement is*

$$E[T_{GPS}] = \sqrt(\pi)\Delta T_{GPS}$$

Proof of Theorem 1: We assume that the number of time slots necessary to acquire a valid GPS measurement is a Weibull random variable with parameters $\alpha = 1$, $\beta = 2$.

$$\begin{aligned} E[T_{GPS}] &= \Delta T_{GPS} \int_0^\infty 2te^{-t^2} dt \\ &= 2\frac{\sqrt(\pi)}{2} \Delta T_{GPS} \\ &= \sqrt(\pi)\Delta T_{GPS} \end{aligned}$$

□

To obtain T_{route} we decompose the time into deterministic components and random times and characterize their distributions. Let t_{WOT} be the amount of time for the module to send an initial "Who's Out There" (WOT) message including back-off time, $t_{WOT-ACK}$ be the minimal fixed acknowledge time to reply with a WOT-ACK message, t_{list} be the time required to add a potential units information to a HUB list, t_{wait} be the module WOT-ACK waiting/listening period, t_{pick} be the minimum time required by module to select a valid HUB, $t_{route-ACK}$ be the time required by module to receive ROUTE HUB ACK, $t_{notify-ACK}$ be the time required to ACK ROUTE HUB recognized, ΔT be the fixed minimal time slot interval for discrete equi-probable random variable, and $T_{backoff}(n)$ be the equi-probable discrete random variable backoff utilized for various interval transmit periods for values $0, 1, \ldots, N$, where N is a positive integer and n is the total number of units attempting to route in a time slot.

THEOREM 2. *The average route time for a single sensor module is*

$$E[T_{route}(1)] = 3t_{WOT} + 2N\Delta T + t_{WOT-ACK} + t_{list} + 3t_{wait} + t_{pick} + t_{route-ACK} + t_{notify-ACK}$$

and for $n > 1$ we have

$$E[T_{route}(n)] = 3t_{WOT} + 4E[T_{backoff}(1)] + t_{WOT-ACK} + t_{list} + 3t_{wait} + t_{pick} + t_{route-ACK} + t_{notify-ACK}$$

Proof of Theorem 2: After a "spark" occurs, the module will execute a WOT of time $t_{WOT} + T_{backoff}(1)$, then wait a fixed amount of time to receive an acknowledgement $t_{WOT-ACK}$, require time to add the list of potential HUBs t_{list}, and reset the wait time for other potential responses t_{wait}. The module then performs a similar process twice more to allow other potential HUBs to respond. Since none will respond in the single module case, the process is deterministic, where a second round WOT occurs with time $t_{WOT} + T_{backoff}(1)$, followed by a waiting time for responses of t_{wait}, then a third WOT of time $t_{WOT} + T_{backoff}(1)$, followed by another waiting interval t_{wait}. Finally, a hub is selected and the HUB is notified taking time $t_{pick} + T_{backoff}(1)$. The module takes $t_{route-ACK}$ to receive an acknowledgement from the chosen HUB and then requires $t_{notify-ACK}$ to complete the transaction with chosen HUB.

We first calculate $E[T_{backoff}(1)]$. Since, $T_{backoff}(1)$ is an equi-probable r.v. for values $0, 1, \dots N$, we have the probability of selecting a time slot $\frac{1}{N+1}$ and fixed time slots of interval ΔT. Thus, we have

$$\begin{aligned} E[T_{backoff}(1)] &= \sum_{i=0}^{N} i \frac{1}{N+1} \Delta T \\ &= \frac{N(N+1)}{2} \frac{1}{N+1} \Delta T \\ &= \frac{N \Delta T}{2} \end{aligned}$$

Taking the expected value of $T_{route}(1)$, we have

$$\begin{aligned} E[T_{route}(1)] &= t_{WOT} + E[T_{backoff}(1)] + t_{WOT-ACK} + t_{list} + t_{wait} + t_{WOT} + E[T_{backoff}(1)] + t_{wait} \\ &\quad + t_{WOT} + E[T_{backoff}(1)] + t_{wait} + t_{pick} + E[T_{backoff}(1)] + t_{route-ACK} + t_{notify-ACK} \\ &= 3t_{WOT} + 4E[T_{backoff}(1)] + t_{WOT-ACK} + t_{list} + 3t_{wait} + t_{pick} + t_{route-ACK} + t_{notify-ACK} \quad (2) \\ &= 3t_{WOT} + 2N\Delta T + t_{WOT-ACK} + t_{list} + 3t_{wait} + t_{pick} + t_{route-ACK} + t_{notify-ACK} \quad (3) \end{aligned}$$

For $n > 1$ the result is similar to the Equation 2 above.

□

EXAMPLE 3. *For the single unit case, we have (all in milliseconds)* $t_{WOT} = 150$, $t_{WOT-ACK} = 2100$, $t_{list} = 560$, $t_{wait} = 3550$, $t_{pick} = 130$, $t_{route-ACK} = 660$, $t_{notify-ACK} = 100$, $\Delta T = 175$, $N = 19$. *Utilize Theorem 2, we have* $E[t_{route}(1)] = 21300$ *ms or rather an average route time of* 21.3 *sec.*

For n sensors where $n > 1$, we need to consider the possibility of a collision occuring in transmission of a packet. Let q be the probability that a module attempts to transmit a message in a time slot. Thus $1 - q$ is the probability that a module does not attempt to transmit a packet in a time slot. For a module to successfully transmit a packet in the first time slot competing against $n - 1$ other modules, we get $q(1-q)^{n-1}$. Let p be equal to successfully transmitting a packet and $1 - p$ is an unsuccessful tranmission of a packet. i.e. $p = q(1-q)^{n-1}$ and $1 - p = 1 - q(1-q)^{n-1}$. Let $p(t)$ be equal to a module successfully transmitting in time slot t with a total of n modules competing to transmit. We then have

$$p(t) = (1-p)^t p$$

for all $t = 0, 1, \dots$.

Let $\hat{N}_{route}(n)$ be the average number of time slots required to send a message from the n^{th} module with $n - 1$ modules competing to send a message. Then

$$E[\hat{N}_{route}(n)] = \sum_{t=0}^{\infty} t p(t)$$

THEOREM 4. *The expected number of time slots required to successfully tansmit a packet with a total of* n *modules competing to transmit is*

$$E[\hat{N}_{route}(n)] = \frac{1}{\frac{1}{N+1}(\frac{N}{N+1})^{n-1}} - 1$$

Proof of Theorem 4:

Recall that q is the probability of attempting to transmit in a time slot. Since each round, time slots are partition equi-probable from $0, 1, 2, \ldots, N$, we have $q = \frac{1}{N+1}$. Let $z = 1 - p$ and we have

$$
\begin{aligned}
E[\hat{N}_{route}(n)] &= \sum_{t=0}^{\infty} tp(t) \\
&= \sum_{t=0}^{\infty} tz^t p \\
&= p \sum_{t=0}^{\infty} tz^t \\
&= pz \frac{\partial}{\partial z} \sum_{t=0}^{\infty} z^t \\
&= pz \frac{\partial}{\partial z} \frac{1}{1-z} \\
&= pz \frac{1}{(1-z)^2} \\
&= \frac{p(1-p)}{p^2} \\
&= \frac{1}{p} - 1 \\
&= \frac{1 - q(1-q)^{n-1}}{q(1-q)^{n-1}} \\
&= \frac{1}{\frac{1}{N+1}(\frac{N}{N+1})^{n-1}} - 1
\end{aligned}
$$

□

THEOREM 5. *Using Theorem 4, the time required to establish n routes is*

$$
\begin{aligned}
E[T_{route}(n)] &= 3t_{WOT} + 4\left\{(N-1)\Delta T \sum_{i=2}^{n} (\frac{N+1}{N})^{i-1}\right\} - 4(n-1)\Delta T + 2N\Delta T + t_{WOT-ACK} + t_{list} \\
&\quad + 3t_{wait} + t_{pick} + t_{route-ACK} + t_{notify-ACK} \ .
\end{aligned}
$$

Proof of Theorem 5:

Recognize that $T_{backoff}(n)$ is equivalent to

$$T_{backoff}(n) = \sum_{i=2}^{n} \hat{N}_{route}(i)\Delta T + T_{backoff}(1)$$

and so using Theorems 2 and 4, we have

$$E[T_{backoff}(n)] = \sum_{i=2}^{n} E[\hat{N}_{route}(i)]\Delta T + E[T_{backoff}(1)]$$

$$
\begin{aligned}
&= \Delta T \sum_{i=2}^{n} \{ \frac{1}{\frac{1}{N+1}(\frac{N}{N+1})^{i-1}} - 1 \} + \frac{N\Delta T}{2} \\
&= (N-1)\Delta T \sum_{i=2}^{n} (\frac{N+1}{N})^{i-1} - (n-1)\Delta T + \frac{N\Delta T}{2}
\end{aligned}
$$

Thus, we have

$$
\begin{aligned}
E[T_{route}(n)] &= 3t_{WOT} + 4E[T_{backoff}(1)] + t_{WOT-ACK} + t_{list} + 3t_{wait} + t_{pick} + t_{route-ACK} + t_{notify-ACK} \\
&= 3t_{WOT} + 4\left\{ (N-1)\Delta T \sum_{i=2}^{n} (\frac{N+1}{N})^{i-1} \right\} - 4(n-1)\Delta T + 2N\Delta T + t_{WOT-ACK} + t_{list} \\
&\quad + 3t_{wait} + t_{pick} + t_{route-ACK} + t_{notify-ACK} \ .
\end{aligned}
$$

$\square$

4.2. Test-bed Results

Now we examine actual results obtained by testing the WOT protocol on a DSSN. For this test, 6 sensor modules as described in Section 2 were built, Fig. 3. One module was used as Home terminal and the other units were utilized as sensor Modules or sensor Hubs.

4.2.1. GPS results

Statistics were obtained for acquisition time of a GPS unit. This is critical to obtaining overall time required to route assuming we consider the entire time from sensor deployment to establishing the complete network topology. A plot of GPS acquisition times is compared to the Weibull random variable with parameters $\alpha = 1$, and $\beta = 2$. With our specific hardware design, typical GPS position acquisition times range from 100 seconds to 200 seconds. This time is greatly dependant on time of day and antenna field of view. For this reason, occasional GPS acquisition times will stretch to several minutes. When deploying a sensor network using the GAaRP protocol, a user must be aware of these factors for deciding when to send the initial RF Spark message from a Home terminal.

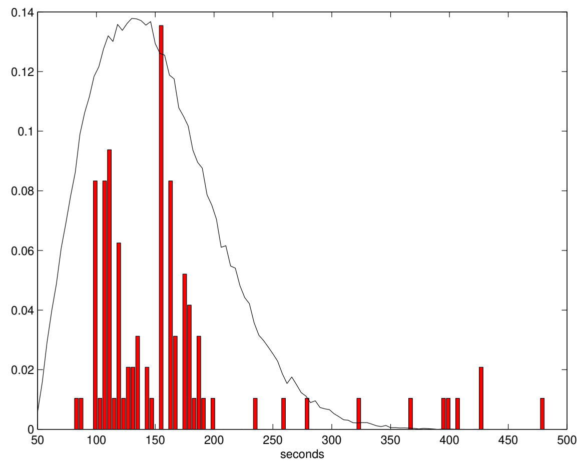


Figure 6: GPS acquisition times

4.2.2. Single Hop Case

In the single hop case, we performed actual field tests to examine the average time required to route n units for $n = 1, 2, \ldots, 5$. As expected the time increases as n increases, on a logorithmic scale. Figure 7 shows a plot of empirical versus theoretical routing times. Although offset, both plots follow a similiar trend. We credit the longer empirical times to indeterminant delays in the compiled software in the COTS RF unit.

Looking at the theoretical equation for single hop routing times, Thm. 4, we see the time to route n units will *blow up* as n gets large. This proves there is a maximum number of units allowed to create a Link Route, through a single Hub unit, given a specific time threshold. Fortunately, it is more likely for the dispersement of sensor units, upon deployment into the field, to create multiple hop topologies.

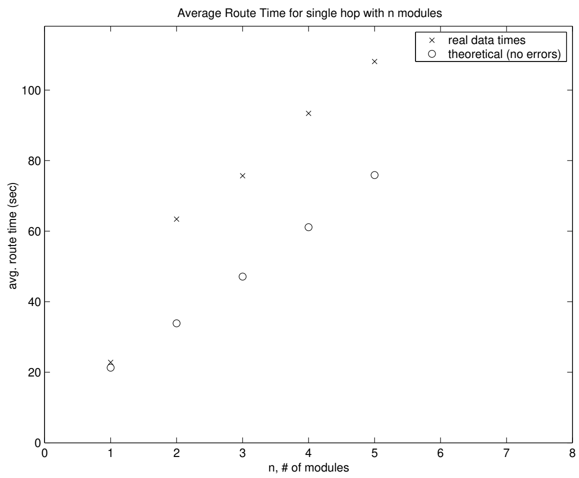


Figure 7: Single hop route for n sensor modules

4.2.3. Multiple Hop Case

To test routing times for multiple hop network topologies, we started with the base case of one sensor unit per hop. With 5 sensor units in the field, this gives 5 hops to Home. For each hop, there are no competing units, so routing times are expected to be linear, starting with the average time to route one unit: 22.8 seconds.

Testing of the GAaRP algorithm is still in progress, so complete test results of other multiple hop network topologies is not available. Preliminary tests show promising results. It was found, in multiple hop scenarios, as more sensor units established Link Routes, the remaining units had greater chances of establishing a Link. In no case was a sensor unit left without a route.

5. STATUS AND FUTURE WORK

The project involves extensive field testing of off-the-shelf equipment, integrated with an innovative routing protocol. Currently, six prototype modules have been placed together with some level of self-routing connectivity tested. These test results are to date, preliminary, but convey a good understanding of the characteristics of the routing protocol. Clearly, we must continue developing, testing, and implementing our routing algorithm to prove the GAaRP protocol as a useful and effective routing scheme. Refinement and confirmation of accuracy of our preliminary analysis will progress as our algorithms are further tuned and optimized. Plans to incorporate computer simulations will strengthen reliability of results from emprical testing.

Thus far, we have proven that a fairly complex routing scheme can be implemented using inexpensive off-the-shelf equipment for the appplication of large scale distributed ad-hoc networks.

REFERENCES

[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.] D. P. Bertsekas and R. G. Gallager. *Data Networks,*. Prentice Hall, 2nd edition, 1992.

[3.] M.E.C. Bowen and G.B. Smith. Design of flexible asic's including embedded processors for smart sensor applications. In *IEE Application Specific Integrated Circuits for Measurement Systems*, pages 5/1 – 5/3, 1994.

[4.] Piyush Gupta and P.R. Kumar. The capacity of wireless networks. *To appear in IEEE Information Theory*, 1998.

[5.] T.C. Henderson, M. Dekhil, S. Morris, Y. Chen, and W.B. Thompson. Smart sensor snow. In *Proceedings of 1998 IEEE/RSJ international Conference on Intelligent Robots and Systems*, 1998.

[6.] E. Jacobsen. The building blocks of a smart sensor for distributed control networks. In *Northcon 1996*, 1996.

[7.] D.B. Johnson. Routing in ad hoc networks of mobile hosts. In *Proceedings of 1994 Mobile Computing Systems and Applications*, 1995.

[8.] R. Morris and B. Karp. Greedy perimeter state routing (gpsr). *Draft, http://www.eecs.harvard.edu/ karp/gpsr4-98.ps*, apr 1998.

[9.] P.Gupta and P.R. Kumar. A system and traffic dependent adaptive routing algorithm for ad hoc networks. In *Proceedings of the 36th IEEE Conference on Decision and Control*, Dec 1997.