PATH PREFERENCE IN SELF-HEALING ROUTING VERIFIED AND IMPROVED THROUGH VISUALIZATION IN SENSE

By

Christopher Frank Morrell

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Major Subject: COMPUTER SCIENCE

Approved:

Boleslaw K. Szymanski, Thesis Adviser

Rensselaer Polytechnic Institute
Troy, New York

November 2008
(For Graduation December 2008)

CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENT

I would like to thank my wife and children for their continued understanding and support throughout my time at RPI. There have been many very long days and many very late nights, and they have seen me through to the end. I'd also like to thank Professor Boleslaw Szymanski for his support and guidance. He was always able to point me in the right direction, and continued to push me to go further and learn more than I thought I could. Lastly, I'd like to thank everyone that I've worked with while conducting this research. Their assistance and insight has been invaluable, and I could not have completed as much as I did without them. Specifically, thanks to Tom Babbitt, Joel Branch, Mark Lisee, Kamil Wasilewski, Sahin Cem Geyik, and Wang Zijian.

ABSTRACT

The problem of reliable and efficient routing in wireless sensor networks is an open and formidable one. Wireless nodes use broadcast as their communication primitive, and radio waves as their transmission medium. These add inherent difficulties in the ability to maintain routing within a network. Broadcast adds difficulty due to the fact that there is no one-to-one connection between nodes, and radio communication often leads to unreliable links. This thesis presents a novel routing algorithm entitled Self Selecting Reliable Path Routing (SRP), which was biologically inspired, and provides a robust yet simple solution. SRP makes an attempt to overcome the problems that are a part of routing in a wireless sensor network. SRP seeks to find a balance between dynamic routing protocols such as GRAdient Broadcast (GRAB), Self Selective Routing (SSR), and Self Healing Routing (SHR), and static routing protocols such as Advanced On Demand Vector Routing (AODV).

SRP has grown from other dynamic routing protocols within the family of Self Selecting Routing Protocols, which began with SSR and SHR. SRP has been inspired by the idea of ants using pheromone to mark paths as they search for paths to food sources, and seek to reuse reliable paths once they are established. In much the same way, once SRP discovers a reliable path from source to destination, it will continue to use that path until such time as the path is broken. If the path is broken, SRP is also robust enough dynamically to find an alternate path to the destination without any central control.

Additionally, researchers are continually looking for new ways to look at different problems. In the field of network protocol research, simulations are often used to design, troubleshoot, and improve protocols. Often these simulations result in cryptic textual outputs that require much effort to understand. If a tool existed that could take the cryptic output and convert it into an animation of a network, it would simplify the work of researchers when trying to understand what is happening in a simulation. This tool also would provide assistance in teaching wireless sensor network principles, by making it easier for students to understand what happens in

the life of a network. We have discovered just such a tool, entitled iNSpect, and implemented it in conjunction with SENSE, our open source wireless sensor network simulator. The combination of these tools provides researchers with the ability to visualize the networks that they are simulating, while leveraging the state of the art in both network visualization tools and wireless sensor network simulation.

1. Introduction

Wireless sensor networks are composed of a large number of nodes equipped with radios for wireless communication, sensors for sensing the environment and CPU's for processing applications and protocols. A significant number of wireless sensor networks consist of battery-powered nodes that are able to operate unattended. Such networks require autonomy of management (self-management), fault-tolerance, and energy-efficiency in all aspects of their operation. These properties are especially important for routing, since multi-hop communication is a primitive wireless sensor network operation that is fault-prone as well as energy-intensive. For instance, commonly observed in such networks are faulty (or, potentially subverted) nodes and transient and asymmetric links caused by wildly oscillating packet reception quality. Faulty nodes and transient links cause severe packet loss and spontaneous network topology changes [1, 2]. In terms of energy usage by sensor network node components, radio operation is typically the most costly, as evidenced by a study in [3] and typical hardware specifications given in [4].

A traditional approach to multi-hop routing is to use routing tables that indicate the neighbor to which a packet should be forwarded to reach a destination; prominent examples include AODV [5] and Directed Diffusion [6]. This fundamental approach, which emulates traditional wired network communication, naturally requires nodes to constantly maintain individual neighbor's states (e.g., active or sleeping) to support routing decisions. In operating conditions typical for wireless sensor networks, such maintenance often requires significant overhead, especially if fault-tolerance is to be supported. Hence, providing efficient routing protocols that naturally accommodate and perform well in fault-prone conditions is still an open and formidable challenge.

This thesis presents the biologically inspired family of Self Selective Routing (SSR) protocols [7], which has been extended with a new protocol, entitled Self Selective Reliable Path Protocol (SRP). SRP introduces preferred path selection, which is the primary research contribution of this thesis, and was introduced in [8].

In SRP, after a node currently possessing a packet transmits it, all nodes that receive it decide which one will forward it. This decision is made autonomously by each receiver based on their respective hop distances to the destination using a transmission back-off delay to resolve potential ties.

This thesis discusses a novel mechanism used by SRP which is a preferred path selection algorithm. This method allows the node that forwards the current packet to select itself for forwarding the next packet in the flow with essentially no delay. This creates a protocol that is both delay efficient (minimal delay to forward a packet in a normal case) and robust (another node will forward a packet if the preferred node is down or has lost its link to the sender) at the same time.

There are other protocols that, like SRP, route on the premise of avoiding neighbor state maintenance and letting receivers contend for forwarding packets. However, they all require geographical location information, which SRP does not. Three such protocols, GRAd [9], GRAB [10], and BLR [11] are not capable of a route repair. GRAB also uses a more aggressive fault-tolerance technique by allowing redundant packets to follow multiple paths to a destination. SRP forgoes this approach and relies strictly on its prioritized transmission back-off delay technique to support (limited) fault-tolerance. Other protocols, GeRaF [12], IGF [13], PSGR [14] and SIF [15] define eligibility regions for packet forwarding and therefore require detailed knowledge of geographical placement of currently active nodes, which is difficult to obtain and maintain in wireless sensor networks.

In addition to the work on implementing a path preference routine in SRP, this thesis also presents work on providing SENSE with a robust visualization capability. Using simulation to debug protocols and algorithms for sensor networks often results in the designer being confronted with pages upon pages of simulator output, that without serious effort to understand, is mostly meaningless. The author in [16] also agrees that we have to find a way to deal with large amounts of information.

> The amount of information available to scientists from large-scale simulations, experiments, and data collection is unprecedented. In many instances, the abundance and variety of information can be overwhelming.

This author also discusses the scientific importance of abstracting data in such a way that it is meaningful to the human brain. Certainly the network setup and statistical outputs are meaningful, but the line by line textual output of which packets went through which nodes in the network does not mean much without a picture. In addition to understanding output, there are some problems that may arise while developing protocols and algorithms for networks, that would not be identifiable without a way to really see what is happening in a network. Without the aid of visualization, the designer would regularly need to analyze pages of simulator printouts, to draw a picture of what happened in the simulation, and where something went wrong. Protocols for wireless networks are particularly sensitive to environment perturbation and transient failures. As a result, any, even small, change in such protocol design may result in huge changes in behavior. Understanding and controlling the ramifications of such changes are greatly aided by the visualization of the simulated execution. Hence, the motivation for visualization work in this thesis is in full agreement with the views of the authors of the nam tool [17] supplementing ns2, who succinctly stated their motivation for developing a visualization tool as follows:

> Protocol design requires understanding state distributed across many nodes, complex message exchanges, and with competing trac. Traditional analysis tools (such as packet traces) too often hide protocol dynamics in a mass of extraneous detail...nam [is] a network animator that provides packet-level animation and protocol-specific graphs to aid the design and debugging of new network protocols...Nam now integrates traditional time-event plots of protocol actions and scenario editing capabilities.

Equally important is the aid that visualization can provide in understanding and teaching how the sensor network operates and how protocols and algorithms designed for sensor networks really work. Particularly in wireless networks, routing and distributed computing is often a complex and dynamic process which is difficult to conceptualize or grasp intuitively and therefore difficult to teach. Visualization can be effectively used to build intuition about the network protocol modus operandi

which then can create a foundation for deeper understanding of the protocol. The use of visualization in teaching networking has been discussed in [18]. The above mentioned debugging and educational needs motivated the visualization research presented in this thesis.

SENSE is our simulator of choice when working on problems involving wireless sensor networks. Recent work consists primarily of the development of routing protocols within these networks. While SENSE is an extremely robust and easy to use simulation tool, the visualization tools provided with it are lacking. In the search for a solution, many iterations of tools have evolved, until a pre-built tool, intended for use with ns2 [19], was discovered. This tool, entitled iNSpect [20], was written by a group of researchers at Colorado School of Mines. When used in conjunction with SENSE, iNSpect provides the ability to easily create an animated playback from the output of a wireless sensor network simulation. This animation provides researchers the ability to step through a simulation, and gain a visual image as to how packets traverse the network, how nodes work together, and how network flows relate to each other. This coordination between SENSE and iNSpect, has been a breakthrough in our use of the SENSE simulator, and has proven invaluable in the advancement of our research.

The remainder of this thesis is organized as follows. Chapter 2 describes the SENSE simulator, giving some background into our simulator of choice. Chapter 3 discusses the historical work regarding the family of Self Selecting Routing (SSR) Protocols. Chapter 4 presents this thesis' primary contribution to this research topic in the form of the newest SSR protocol, Self-Selecting Reliable Path Protocol (SRP). In addition, chapter 4 presents comparisons between SRP and SHR, AODV, GRAB, and tests from an implementation of SRP on Motes sensors. Chapter 5 addresses this thesis' secondary research contribution, which is the implementation of a visualization tool in conjunction with SENSE. Lastly, Chapter 6 presents conclusions and possible future work that could stem from this research.

2. SENSE

SENSE is a C++ based simulator which was created to specialize in the simulation of wireless sensor networks. Originally presented in [21], SENSE was created to provide extensibility, reusability, and scalability to wireless network simulation. SENSE uses a component-port model to represent parts of a network, and is built on top of COST [22], which is a general purpose discrete event simulator. COST is built on top of a component based C++ extension entitled CompC++ [23].

By using a component-port model as the basis of design, SENSE achieves its goal of being extensible, as new protocol stacks are easily implemented. This model allows a user to abstract many of the details of the simulator, and simply create new network components as needed. In the case of a wireless sensor network, the sensor is a composite component, that consists of several lower level components. These low level components include the layers of the protocol stack, mobility, and power management.

In SENSE, components are connected by two types of ports, inports and outports. Inports are generally functional in nature, in that they implement a certain function. Outports on the other hand can be described as an abstraction of a function pointer; they define what functionality they expect of others. More specifically, the ports are used to connect components. For example, each layer of the protocol stack, which is defined as a component, gets an outport pointing up to the next higher layer, and an outport pointing down to the next lower layer. In addition to these, each layer also has an inport coming from each layer around it. Layers that are required to communicate directly with the mobility or power management components have additional ports to allow data to flow from those components into the protocol stack. Inports and outports are simply connected to one another, providing a connection between components for data to flow through and information to be exchanged.

In addition to providing the ability to easily expand the simulator to support newly developed protocols, SENSE was written to include support for many popular

protocols. These included protocols range all the way from the physical layer to the application layer, and include IEEE 802.11, AODV, several radio models, several power management models, and others. Originally SENSE was created as a wireless sensor network specific response to the wildly popular network simulator ns2 [19].

As popular as ns2 is, without many add-on packages, it is not well suited to simulate wireless sensor networks. The fact that ns2 was originally written as a wired network simulator, the layers required to simulate a wireless network in this environment increases its complexity. Since its first publication and use in simulating SSR, SENSE has gained a following throughout the world, as evidenced by a Google search of "simulating results of WSNs," which reported 36 papers. For example, in [35] the authors simulate C^2E^2S (Cluster and Chain based Energy Delay Efficient Routing Scheme) for wireless sensor networks. In [24], the authors used SENSE to simulate Coordinate-based Data Dissemination protocol (CODE) and Sink Cluster-based Data Dissemination protocol (SIDE). In addition to these, SENSE was used as the simulator by [25] and [26].

3. Self Selective Routing Protocols

There is a fundamental difference between wireless and wired networks because the basic communication mode is broadcast in the former and point-to-point in the latter. Self-Selection [27] takes advantage of broadcast communication to efficiently implement the basic operation of selecting a node possessing some desired properties among all the neighbors of the requester. Self-selection employs a prioritized transmission back-off delay scheme in which each node's delay of transmitting a signal is a measure of the node's fitness to perform a pertinent task and in turn, enables the node to autonomously select itself for the task. This ability to perform self-selection has become the basis for our protocols.

SSR has become a family of protocols, each with different reliability level and all based on the same principal of self-selecting the route at each hop of a multi-hop path. This chapter compares the basic members of this family, called Self Selective Routing (SSR) and Self-Healing Routing (SHR), and is followed by a chapter detailing the most recent addition to the family, Self-Selective Reliable Routing Protocol (SRP). Having evolved from each other, subsequent versions are strongly based on previous versions, and as such the protocols share a basic framework.

3.1 Self Selective Routing (SSR)

SSR consists of two primary stages. The first consists of network discovery, while the second consists of data transmission. In all protocols of the SSR family, each node knows its distance, in the number of hops, from a destination node. Currently, it is assumed that nodes are not mobile, which allows us to use the same hop-count distances throughout the life of the network. This distance is established via the network discovery stage, which consists of an initial route request and route reply by the source and destination nodes in each network flow. In SSR this route request and reply stage stems from a biological inspiration.

Ants use a pheromone to mark paths and communicate information about food sources among different insects of the same colony [28]. The destination request

flooding corresponds to the initial search for food in which ants randomly explore the environments and in the process mark the branching paths with pheromone. Packets sent in this stage are referred to as DREQ (Destination Request) Packets. The destination reply phase corresponds to a walk back to the colony by an ant that found a food source. Walking back, an ant will mark branches on the path home with pheromone to distinguish the return path from others. Packets sent in that stage are called DREP (Destination Reply) Packets. In the SSR protocols, the initial flooding is only conducted once at the sensor network deployment for all potential destinations using the signal-strength aware flooding described in [21].

For the packet forwarding process, instead of only one designated neighbor receiving the packet sent by the sender, all of its neighbors receive it. The neighbor nodes then use the self-selection algorithm to decide autonomously which node will forward the packet. This self-selection algorithm uses a prioritized transmission back-off delay scheme. In this scheme, after a node receives a packet, it sets a timer for a random delay based on its distance, in terms of hops, from the destination. The transmission back-off delay for SSR is specifically determined by the following equation:

$$d_{back-off} = \begin{cases} \lambda \cdot ((h - h_{expected} + 1) \cdot U(0,1) & if h > h_{expected} \\ \frac{\lambda}{h_{expected} - h + 1} \cdot U(0,1) & if h \leq h_{expected} \end{cases} \tag{3.1}$$

h is the node's hop distance from the destination, $h_{expected}$ is the sender's hop distance minus 1 (as in fault tolerant network the best forwarding node should be this distance from the destination), $U(0,1)$ is a real random number uniformly distributed between 0 and 1 (randomizing delays to reduce collisions) and λ is a scaling factor that defines the stretch of random delay values.

Equation 3.1 ensures that the nodes closest to the destination have the highest probability of forwarding a packet. If a node overhears another node forwarding the same packet which it is waiting to transmit, it will cancel its own transmission. Upon hearing the packet being transmitted, the sender will also send an acknowledgment (ACK) packet signaling all nodes within its communication range to cancel their transmissions, just in case the self-selected node's transmission is out of range of

receivers competing to forward that packet. This process repeats until a packet reaches its destination.

SSR's benefits lie in its low overhead (SSR does not require explicit route maintenance or node location information) and fault-tolerance, since packets are received over all links of the sender and therefore have a high probability of reaching the best available neighbor in each transmission. However, SSR suffers from two limitations.

First, delays based on Equation 3.1 result in packets unnecessarily traveling longer routes even if shorter routes are available. If there are no failures in the network, then it is clear from the way the hop count to the destination is established that each node has at least one neighbor that is one hop closer to the destination than itself. It is also clear that all neighbors must have their hop distances within a small range of the sender. Namely their distances must be at most by one smaller and at most by one greater than its hop distance. The delays generated according to Equation 3.1 may result in a neighbor that is farther from the destination than the sender forwarding the sender's packet, therefore routing a packet via a path longer than necessary. For example, consider the network shown in Figure 3.1, where nodes are represented by circles and their hop distances from the destination (labeled DST) are indicated by the numbers in the circles. Suppose that node A has forwarded a packet from the source (labeled SRC) with an expected hop distance of 2, and node B and D *compete* for forwarding it (node SRC will not try to forward the packet since it just sent it). From Equation 3.1, node B's delay will be $d_{B\_back-off} = \lambda \cdot U(0,1)$ and node D's delay will be $d_{D\_back-off} = 2\lambda \cdot U(0,1)$. The probability that node D will choose to forward the packet is then:

$$p = \int_0^{\lambda} \frac{\lambda - x}{\lambda} \frac{dx}{2\lambda} = \frac{1}{4} \tag{3.2}$$

Therefore, node A's packet has a one in four chance of following a route of length 5 instead of 4. The probability of selecting the longer route of course increases if there are more nodes in the sender's neighborhood through which such a route could be traversed. Hence, Equation 3.1 can be improved to reduce such probability p and therefore enable better performance.

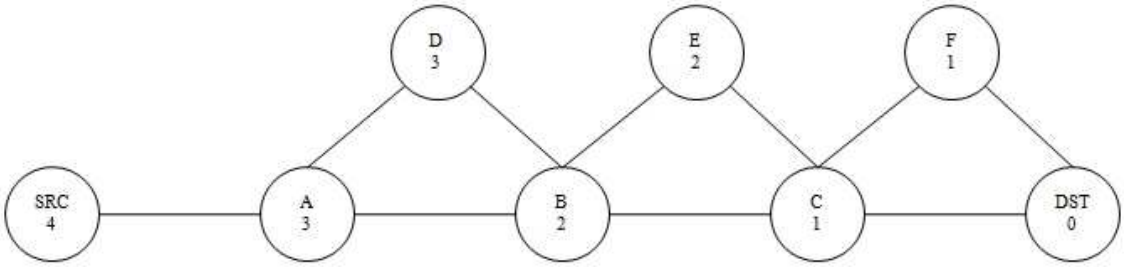

Figure 3.1: Diagram for a packet routing illustration.

The second limitation of SSR is that it does not support any route repair routine for propagating packets around severed routes, which occur when, for a particular node, all its available neighbors have higher hop distances to the destination than itself. Currently, upon encountering a severed route, a packet may by chance travel backwards towards its source until a new route is found in a way similar to the scenario in Figure 3.1. Relying on such backward travel is inefficient. First, probability of subsequent backward hops drops exponentially with the number of hops, so it is very likely that packet will exceed its time-to-live counter before it reaches the destination in such situation. Additionally, SSR will not adapt its behavior in such a way as to prevent further packets from traveling down the severed route to the cut-off point. These shortfalls in SSR prompted the development of Self-Healing Routing (SHR).

3.2 Self Healing Routing (SHR)

The primary difference between SSR and SHR is the implementation of a route repair, i.e. healing, routine. First, upon receiving a DATA packet, instead of using Equation 3.1, a node will ignore the packet if its hop distance is larger than the expected hop distance of the packet plus retransmission bit. Otherwise, it will use the following equation to determine the delay before forwarding the packet:

$$d_{back-off} = \frac{\lambda}{h_{expected} - h + 1 + retransmission} U(0,1) \tag{3.3}$$

As the name indicates in Equation 3.3, retransmission is 0 for the regular DATA packets or packets sent in the route repair step and 1 for packets retransmitted

during the resending stage (described later). As in the case of Equation 3.1, delays computed according to Equation 3.3 ensure that those nodes that are closer to the destination than the sender forward their packets before those that are not. Additionally, Equation 3.3 generates delays for nodes that are no closer to the destination than the sender only if there are no responses from the nodes that are closer. Hence, no packet will travel a route longer than necessary.

The second improvement is the addition of a route repair routine for propagating packets around severed routes. As previously mentioned, a severed route occurs when a sending node has neighbors that are all farther from the destination than itself. In this case, corrective action must be taken to reroute packets along the remaining shortest route.

The route repair routine is established so that a node will attempt to forward the packet two times. If at that point it fails to do so, a packet is sent with the hop count to the destination increased by two and the node's stored hop count for the flow is increased by two. This has two effects. The first is an attempt to reroute the packet locally. The second is to prevent the node from winning future competitions to forward a packet along the affected flow.

An example of the route repair routine is given in Figure 3.2, which shows how the route repair scheme works to quickly fix the blocked route. Suppose that node D is either asleep or down and node C has a packet to transmit as shown in Figure 3.2(a). Lack of response to node C's second transmission will cause node C's hop distance to increase to 4 as shown in Figure 3.2(a). When the next packet of the same flow is received by node B, its transmission and retransmission will not have responders, so node B will increase its hop distance to 5 as shown in Figure 3.2(b), the packet then will transmit to node C and it will again transmit and retransmit unsuccessfully, so node C will increase its hop distance to 6 as shown in Figure 3.2(c). The next packet received by node A will not be able to transmit, so node A will increase its hop distance to 6, and trigger transmission of the packet to nodes B and C, increasing their distances to 7 and 8, respectively (see Figure 3.2(d)). In this scenario, the next packet from the source will find the only alternative route via nodes E, F, G, and H, completing the route repair and sending this packet on the

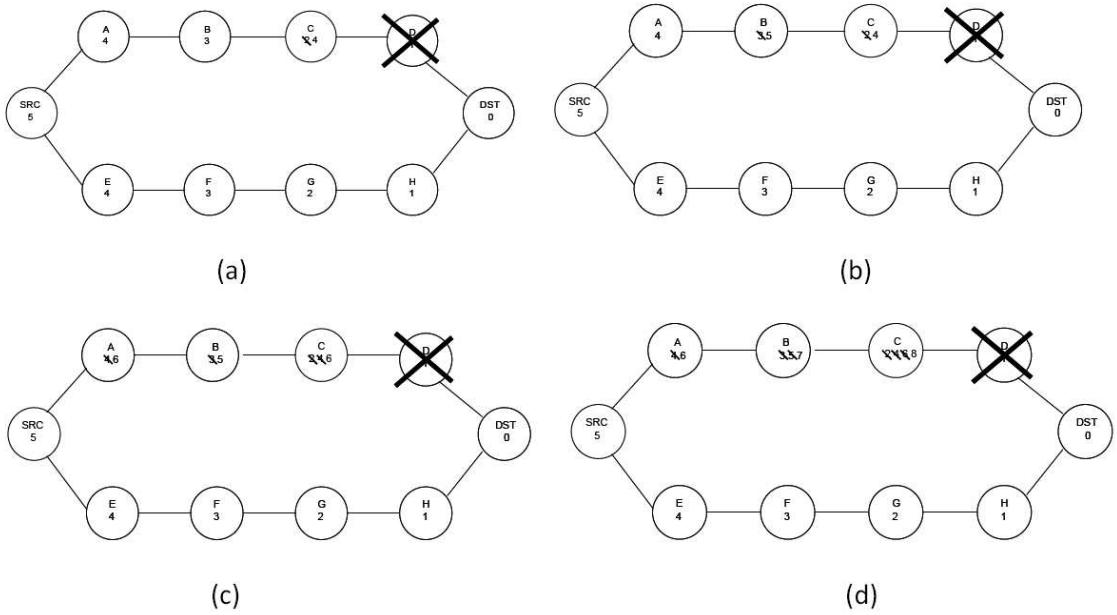

Figure 3.2: SHR Route Repair Scenario

route to the destination. From this point on all packets will travel along the new path.

Although the route repair was initially reported in [7], its costs or even convergence was not established. In [38], an upper bound was established on the cost of route repair in SHR. As already described, in SHR the sender of a packet listens to the response to its transmission. If such a response does not arrive within the time λ, signaling the failure of the previously existing link, the node retransmits the original packet. After the predefined number of unsuccessful retransmissions (two in the current implementation), the sender increases its distance to the destination by 2, as lack of responses to the transmission and retransmissions demonstrates that the only surviving neighbors are nodes with hop distance at least one larger than the current hop distance of the sender. We call such a step a recalibration of the hop distance. Let's consider a sensor network of n nodes in which there is a failure of nodes or their links after which the shortest path from the source to the destination surviving the failure is of length $l < n$. That means that once all nodes not on any of the surviving paths recalibrate their distance to at most n, and the nodes on the surviving paths recalibrate to their correct value, also at most n, then all

traffic will flow through the shortest surviving path. The smallest initial distance that nodes needing recalibration might have is 1, so at most $(n-1)*n/2$, hence $O(n^2)$ recalibration steps are needed.

4. Self Selecting Reliable Path (SRP)

4.1 SRP Background

In our research, we found that SSR and SHR worked well in networks with a high failure rate. This was due to the fact that packets could take any number of paths from source to destination. The problem was that the protocols failed to capitalize on a network that was relatively reliable. This is the problem that we solved in the implementation of Self Selecting Reliable Path Protocol (SRP), which was originally introduced in [8]. The inspiration of the ant pheromone analogy described in SSR, has been extended from the network initialization phase into the data transmission phase. Simply, the data transmission phase corresponds to ants following the marked path to the food source, where at each path fork the strength of the pheromone dictates which branch of the fork to select. Therefore, as additional ants use a reliable path to the food source, the strength of the pheromone on that path increases, thus increasing the likelihood that other ants will follow. The pheromone level on unused paths will fade in time, helping to increase the delta between weakly and strongly scented paths, making it easier for ants to identify the desired path. To extend the biological analogy to the protocol, we designed a simple scheme that gives priority to nodes that reside on a reliable path.

In the scheme used to promote reliable links, a preferred path selection was introduced, in which a node that forwarded the current packet will respond almost immediately to a transmission of the next packet in the same flow. To simplify processing, these nodes calculate their delay by dividing the regularly selected back-off delay by 625, while ensuring that it remains larger than the radio transition time. This results in a back-off delay between 20 and 160μs, given λ is 100ms. This minimizing of back-off delay almost guarantees the node future self-selections, thereby stabilizing repeatedly traversed paths. In reference to the slow fading of the pheromone, we have decided to not follow the biological inspiration literally. Instead, the full range back-off delay is immediately restored after the preferred node fails to self-select, as such failure indicates that the recently used node is no

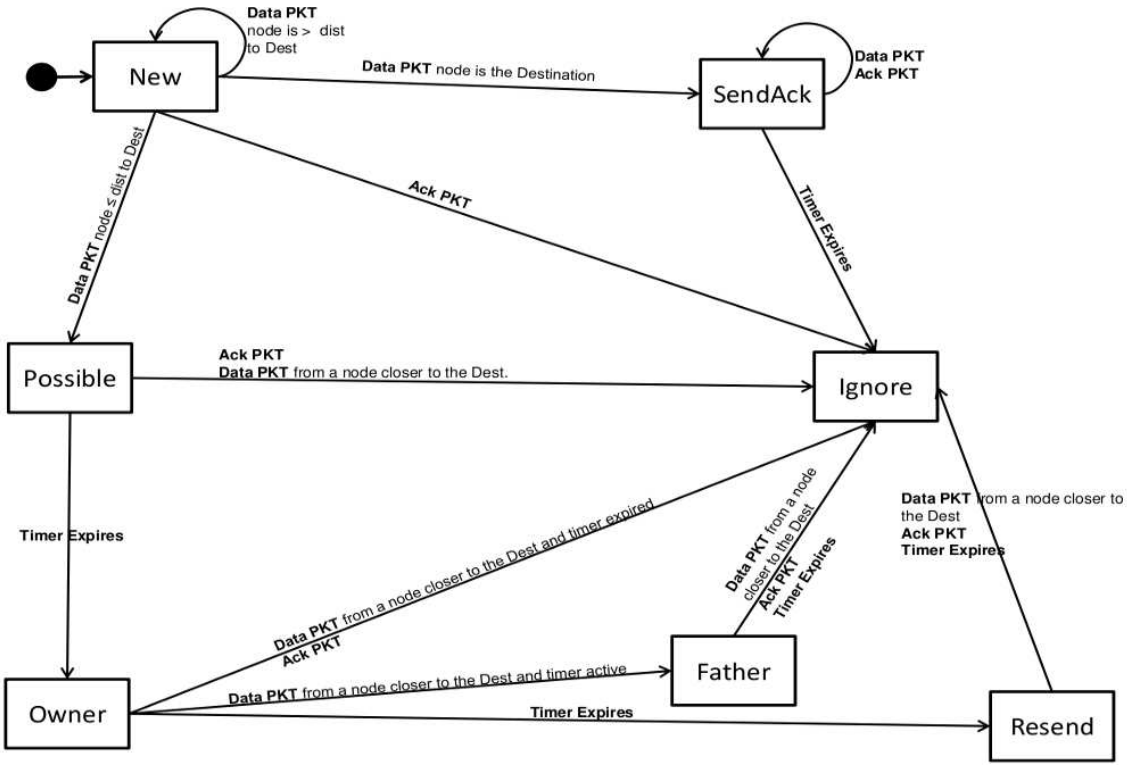

Figure 4.1: State diagram for SRP

longer reliable. Despite its simplicity, the effect of using the preferred path selection in SRP is very positive, as demonstrated in the section below.

Outside of path preference, much of the SRP protocol remains the same as SHR. As shown in Figure 4.1, the data transmission stage can be represented by a Finite State Automaton (FSA) that defines the input, actions and output generated in each state of a node in the network as it routes data (similar FSAs can be defined for the destination request and reply stages). For example, when a node receives a packet that it has not seen before, it immediately moves into the *NEW* state, and depending on its input and status (e.g. data packet received by the destination, data packet received by a node closer to the destination then the sender, acknowledgment packet received, etc) the node transitions itself into the corresponding state and executes the associated actions (for clarity, not shown in the figure).

When the source transmits a DATA packet, only neighbors that are closer to the destination than the sender will react. Depending on the reacting nodes proximity to the destination in relation to the sending node, it selects a transmission

back-off delay. That delay is uniformly distributed between 0 and $\lambda/2$ if the reacting node is one hop closer to the destination. If the reacting node is more than one hop closer, the back-off delay is selected between $3\lambda/4$ and λ. This difference in back-offs ensures that the more reliable single hop closer neighbors have priority over the less reliable multiple hop closer neighbors. λ is a scaling factor that allows us to tune the probability of collision of the nodes' responses. If, during the back-off delay, a DATA packet is received from a node that is closer to the destination, the receiving node cancels the forwarding of the DATA packet and moves to the Ignore state. When the transmission back-off time expires, the node increments the packet's actual hop count by one, sets the expected hop count to its hop distance to the destination and then transmits the packet. After forwarding the packet, the node monitors the carrier to determine if the packet has been forwarded. Lack of forwarding causes retransmissions, and finally route repair which is accomplished by increasing the node or packet's hop distance to the destination by 2 and retransmitting, which is the same scheme described previously as part of SHR.

4.2 SRP Performance Evaluation in SENSE

4.2.1 SRP Compared to AODV and SHR

As originally presented in [8], a large scale network was simulated to compare performance of SRP, SHR and AODV [5]. AODV is representative of traditional route-based routing protocols which find a single best route to the destination, store it in the source or over the route, and use flooding to repair this route when it becomes damaged. It is also typical in its use of acknowledgments to ensure high delivery ratio at the cost of additional packets sent and received during transmission.

The base configuration for the simulations consists of an 8 unit by 8 unit terrain populated with 500 nodes, each with a nominal transmission range of 1 unit. Simulations use the free space propagation model [29]. The simulated application sends packets of a mean size of 1000 bytes at a mean interval of 40 seconds. In each of the several simulations run, we tested the protocols' performance against a change in one of the following test parameters: (1) the rate of permanent node failures; (2) the rate of transient node failures; and (3) the number of sources communicating with a

single destination (base station). SRP and SHR used $\lambda = 100$ms and the maximum hop count equal to the distance to the destination plus $\log 2$ of this distance. We gathered the communication delay at the destination, the packet delivery ratio at the destination and the total number of MAC layer packets transmitted.

4.2.1.1 Single Destination Simulations

The impact of increasing the number of sources communicating with a single destination was also tested, which is a situation that is common in wireless sensor networks. The results of this test are shown in Figure 4.2. Increased traffic causes more random collisions in SHR, decreasing the delivery ratio. AODV maintained a higher delivery ratio at the cost of an increased number of MAC packets produced and larger communication delay. When the number of sources passes 40, AODV must spend so much time maintaining its topology that its performance drops dras-

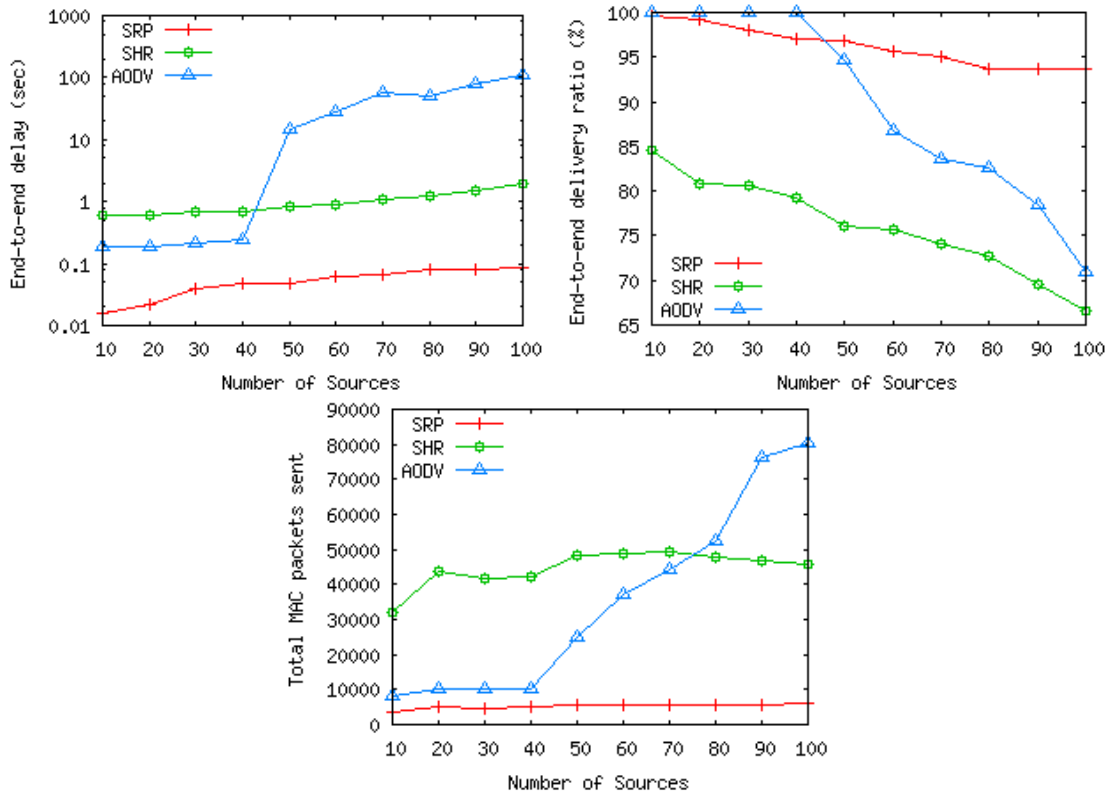

Figure 4.2: Performance of SRP and SHR versus AODV over a reliable sensor network with increasing number of sources reporting to the single base station

tically. SRP on the other hand, maintains an extremely high delivery ratio at very quick speeds despite the large increase in traffic. The huge difference in performance between the SSR family protocols and AODV required the use of a logarithmic scale on the end-to-end delay chart. It is also worth noting that since SRP uses so many fewer MAC packets than AODV, power savings becomes an added, although unintended, benefit.

4.2.1.2 Node Failure Simulations

Two node failure modes were tested, the first to be discussed is permanent failures (see Figure 4.3), followed by transient failures (see Figure 4.4). In sensor networks, transient failures are caused mainly by error-prone links, power management induced duty cycles, and packet collisions. Of these, the duty cycle induced failures are the least disruptive since they are often coordinated with the networking protocol, although this is not the case here. The simulation results presented here are based on a random transient failure model, so they exaggerate the effect of duty cycles on the protocols.

When the topology changes, either by a node failing or returning to the network, extra work is required of the networking protocol. The goal is to minimize this work when the failure is transient, yet quickly update the route when the failure is permanent.

When a single permanent failure was introduced at a fraction of the nodes, both AODV and SRP coped well with the disruption and relatively quickly and efficiently found an alternate route. SRP achieved this with smaller delay and significantly fewer packets than AODV, however with a slightly lower delivery ratio as is seen in Figure 4.3.

In case of transient failures, shown in Figure 4.4, AODV is strongly impacted by topology changes. Link layer failures caused AODV to flood the network looking for a new route. The flooding may stop after a few steps, but it is still disruptive. SRP is affected by transient failures (100% delivery rate drops to 57%) but transmits significantly fewer packets than AODV. As the transient failure rate increases, the failures may overcome SRP's ability to repair routes. A simple solution would be

to simply send each packet twice in a transient failure prone network, which would increase delivery ratio, maintain faster speeds than AODV, and still use significantly fewer MAC packets.

4.2.2 SRP Compared to GRAB

As part of the research presented in [36], we conducted a number of simulations to compare SRP to the published results of GRAB in [27]. These simulations were conducted to show that SRP could compete against another protocol developed specifically for sensor networks. These dynamic protocols use a similar technique in which nodes compete for forwarding the packet at each hop on the way from the source to destination. The design of GRAB is described in [10]. Using SENSE, we conducted a series of simulations to mimic the ones published in [10], which included delivery rate of the protocol as a function of node failure rate and packet loss rate,

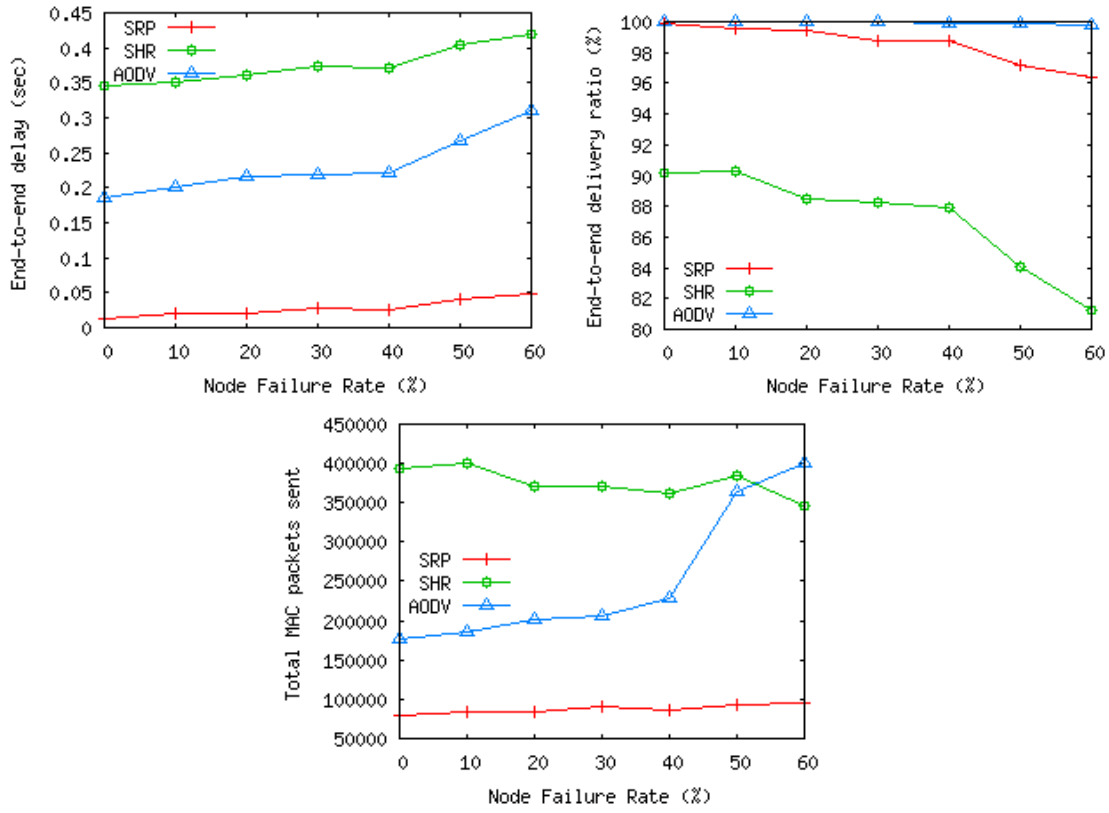

Figure 4.3: Performance of SRP and SHR versus AODV over a sensor network with permanent failures

as well as delivery rate as a function of network density (total number of nodes in the simulated area).

The authors used a 150 meter by 150 meter topology with 1200 nodes uniformly distributed. They simulated a network with one sink and once source node. The source generated a packet every 10 seconds and sent a total of 100 packets. The nodes were an abstraction of the Berkeley motes [34], which consist of an RF Monolithics 916.50 MHz, transceiver (TR1000) radio that broadcasts with 19.2 Kbps of bandwidth. The transmission and receiving time for a packet was 10ms and the transmitting radius of the radio was 10 meters. Both the two ray and free space signal propagation methods were used but only the two ray results were published. There is a footnote that states that the free space signal model gave similar results. The reported results were averaged over 10 simulation runs.

To match the settings under which those results were obtained, we simulated

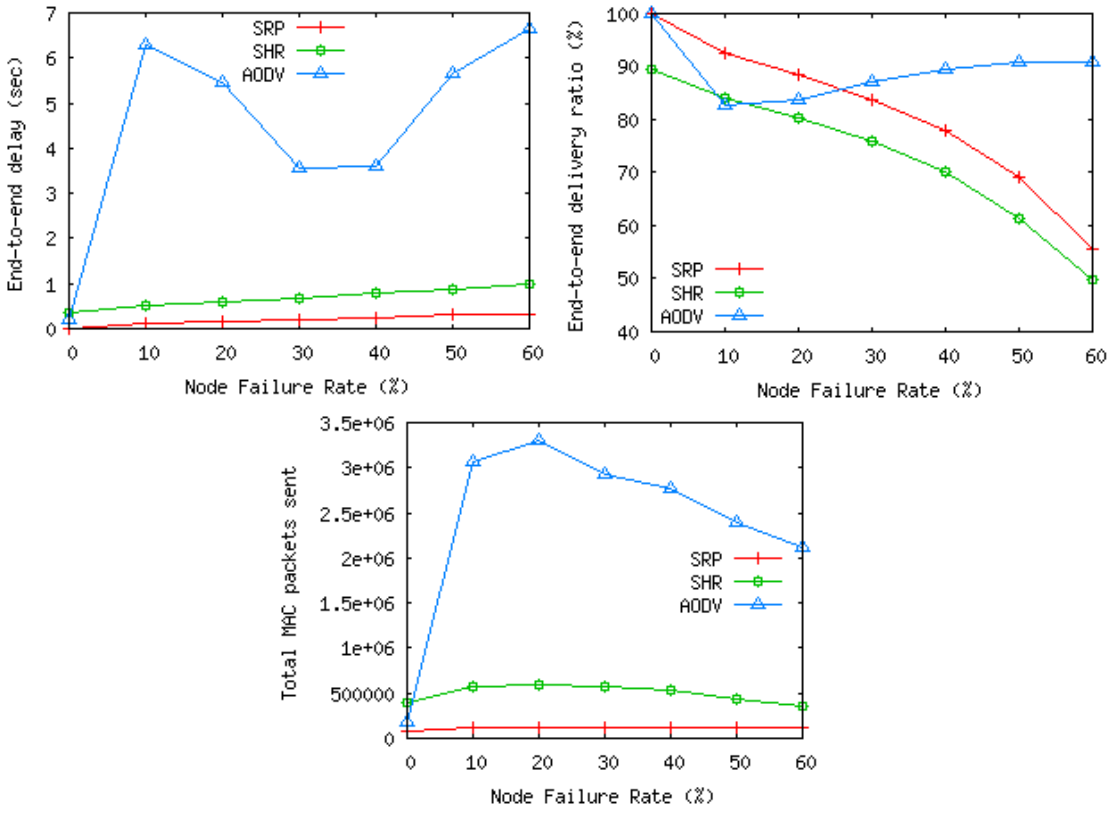

Figure 4.4: Performance of SRP and SHR versus AODV over a sensor network with transient failures

performance of SRP under both the density test and the permanent failure test. 1200 nodes populated a 15 unit by 15 unit terrain, in which each node is stationary, and has a single unit nominal transmission range. Packets were sent every 10 seconds, and simulation ran for 100 packets. Each simulation was executed ten times, each time with a different random number seed. The same 10 seeds were used for all simulation sets. λ was set to 100ms.

For both tests, the authors of [10] used a 15% link failure rate, which they call a packet loss rate, and either changed the permanent failure rate from 0% to 50% in the failure test, or set it constant at 15% for the density test. We used the permanent failure rate functionality of SENSE. To match the experimental measurements collected in [4, 30] for Crossbow MicaZ nodes, we randomly chose 1/6 of the links as unreliable and dropped 90% of the packets that used those links. This amounts to a total of 15% as the link failure rate (that is packet loss rate reported in [10]). In selecting the transient links in our simulation, we have not considered physical distance from the sender. In a real deployment, most transient links are at the far edges of the radio transmission range. Yet, there can easily be some links that are closer to the sender if an obstacle reduces the transmission range in a particular direction. By choosing 1/6 of the links to be transient, and dropping 90% of packets they overhear, we effectively lost 15% of the packets at the node level.

4.2.2.1 Varying Network Density

In the density simulation we set the permanent failure and link failure rate to 15%. Similar to the simulations reported in [10], ten simulations were run for each density level from 600 to 1800 nodes in increments of 200 nodes. The results for the density test show that SRP is considerably more effective than GRAB in sparse network topologies, as depicted in Figure 4.5.

For a node density of 600, GRAB had approximately 36% delivery rate while SRP's was 60.6%. SRP continued to outperform GRAB until the network size reached 1,000 nodes. At that point, the delivery rate for both protocols stays above 95%. The reason that SRP performs well in sparse networks is that it does not restrict the position of the nodes used for forwarding, like GRAB does, and therefore

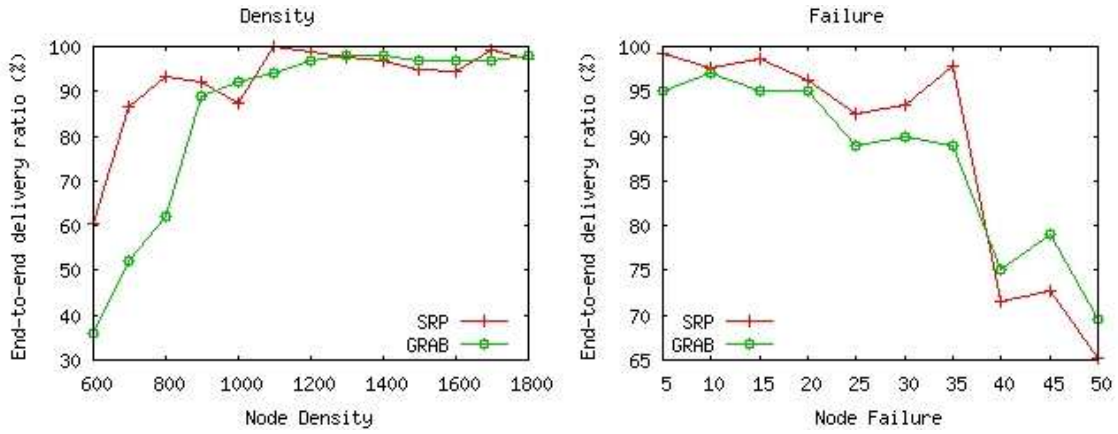

Figure 4.5: Comparison between SRP and GRAB under density and permanent failure tests with a total of 100 packets sent.

will find any available route more readily than GRAB.

4.2.2.2 Varying Network Failure Rate

In the permanent failure simulations the transient link failure rate was set to 15%. Ten simulations were run for each permanent failure rate from 5% to 50% in increments of 5% to get the results comparable to those reported in [10] with the configurations described above. The nodes that failed as part of the permanent failure rate were randomly chosen and failed with probability uniformly distributed over the running time of the simulation. The results for the failure test in Figure 4.5 show that performance of SRP is very comparable to that of GRAB.

At the higher permanent failure rates GRAB does marginally better. At 50% permanent failure rate, GRAB has approximately 69% delivery rate compared to 65.2% rate achieved by SRP. However at 35% failure rate, SRP's delivery rate of 95% exceeded the 89% of GRAB. Both protocols maintain over 95% delivery rate when permanent failures are less than 20%.

SRP attempts to take advantage of both: (1) dynamic route selection similar to the way GRAB and SSR select paths from source to destination, and (2) static routes that quickly push traffic through a stable network. When the permanent failure rate is 40% or higher, SRP is in complete dynamic selection mode especially when considering those node failures that cause considerable turbulence with a test

length of only 100 packets. However, when a semi-stable route can be found, even for a short period of time, the reliable path is quickly established and taken advantage of to speed packets through the network. Existence of such semi-stable routes explains the huge jump in delivery rate for SRP that occurs when the failure rate drops from 40% to 35%. GRAB enjoys a similar jump, but it is not as pronounced. Additionally, when simulations are run longer than for 100 packets the delivery rate of SRP, even with a 50% permanent failure rate, is considerably higher.

5. Visualization

5.1 The History of Visualization Within SENSE

As originally presented in [37], SENSE was originally created without the ability to provide visual feedback from a simulation, and although SENSE has been a commonly used simulator, the lack of visualization has been one of the largest complaints about it [31]. Normal simulation feedback looked much like Figure 5.1. Visual Route (VR) lines provide simulation time, source, destination, packet number, end-to-end packet transmission time, and a node by node path through the network. Assign Sequence Number (ASN) lines are output upon the source identifying a packet to be sent. Data includes packet number, maximum hops to destination, packet type, and simulation time. All of this information is very useful to a researcher who is testing a new network protocol, but the effort required to truly understand what each line means is enormous. One would have to manually plot each node location and overlay each packet's path in order to see how each flow, packet, or node related to each other.

```
VR: 33.300544:  32-> 28: sn00200005: 0.005218: 5:  { 5}  25  26   7  56  28
ASN: sn00270005; max hop 5; DATA; @ 34.310709
VR: 34.314345:  39-> 28: sn00270005: 0.003636: 3:  { 3}  60  48  28
ASN: sn0048000D; max hop 8; DATA; @ 34.613200
VR: 34.619418:  72-> 28: sn0048000D: 0.006217: 6:  { 6}  41  54  66  52  15  28
ASN: sn00100004; max hop 7; DATA; @ 36.225076
VR: 36.230885:  16-> 28: sn00100004: 0.005809: 5:  { 5}  61  66   1  50  28
ASN: sn002E000A; max hop 8; DATA; @ 37.106677
VR: 37.123411:  46-> 28: sn002E000A: 0.016733: 6:  { 6}  32  25  26  40  56  28
ASN: sn0048000E; max hop 8; DATA; @ 37.232596
VR: 37.238826:  72-> 28: sn0048000E: 0.006230: 6:  { 6}  41  54  66  52  15  28
ASN: sn00200006; max hop 7; DATA; @ 39.199971
VR: 39.205174:  32-> 28: sn00200006: 0.005203: 5:  { 5}  25  26   7  56  28
ASN: sn0048000F; max hop 8; DATA; @ 39.851991
VR: 39.858196:  72-> 28: sn0048000F: 0.006205: 6:  { 6}  41  54  66  52  15  28
```

Figure 5.1: An example of the text only output from the SENSE simulator.

As SENSE was improved through several versions, some minimal visualization capabilities were added to it. These new capabilities allowed it to extract a plot that included the locations of all of the nodes as shown in Figure 5.2(a). In addition

to this, SENSE would provide a graphical representation of the path taken by any single packet, shown in Figure 5.2(b).

While this basic level of information is useful, more is still required to fully understand what is happening throughout the lifetime of an entire network, or even throughout the lifetime of a given network flow or packet. Of course, through some basic image manipulation, additional information was available, including the ability to overlay node identities, node locations with paths, multiple paths, and with some effort some animation of packets moving across the network. In order to create the animation, separate images were created of each packet's path on the network, which were then stacked together into an animation. Although this animation could provide additional information, and give a more visually appealing view of the network, it was static, and the time and effort required to build this product was excessive in relation to the information gained from it. This method of visualizing the network also failed to provide any useful statistics with the pictures and animations.

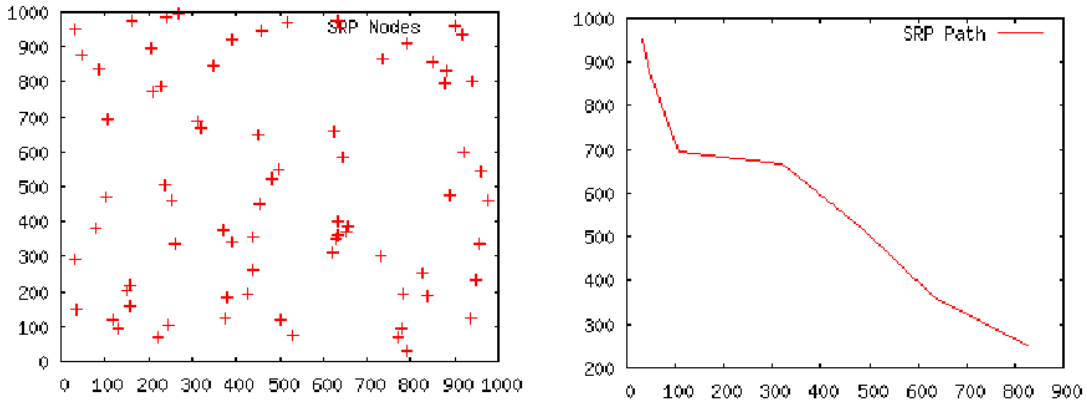

Figure 5.2: (a) An example of the original visualization capabilities built in to SENSE, this plot shows the location of 75 nodes on a 1000 unit by 1000 unit area. (b) The path followed by the first packet in a simulation of the SRP protocol.

5.2 Improving Visualization Within SENSE, Using iNSpect

Although several iterations of tools were created to provide some visualization functionality to SENSE, it was decided that work on the functionality of the simula-

tor would be more important, and implementing and already existing visualization tool with SENSE would be more appropriate. Since ns2 is arguably the most popular network simulator in existence, of course the first choice of visualization tools was the Network Animator (NAM), which is packaged as part of ns2. NAM was made to work with SENSE, but as it was never really designed to visualize wireless networks [32], it is missing key capabilities required in a wireless simulation. Since broadcast is the primary medium of a wireless sensor network, and our protocols are self-selecting, there is no predefined path or link between two nodes. Nam requires a link to be present in order to track packets between nodes. NAM's depiction of a broadcast also did not provide the information desired in a visualization tool for SENSE. In NAM, a broadcast is depicted by a simple circle around the broadcasting node that is the size of the wireless transmission range. In the SSR protocols, the node within that circle that would be selected to forward a packet is not known, hence the link had to be created after the fact, thus increasing the complexity of the model. Although NAM did not meet our specific requirements, it did lead us to the discovery of the visualization tool iNSpect.

Originally presented in [20], a group of researchers at Colorado School of Mines created a visualization tool called iNSpect that is specifically tailored to visualize wireless network simulations. Although intended for use with ns2, and in the future is planned to work well with ns3, we've chosen to use iNSpect as the visualization tool for SENSE. The shortcomings in NAM that have been discussed were also identified by other researchers, and led to the development of iNSpect. iNSpect is a C++ OpenGL based visualization tool that takes ns2 mobility and simulation files as input, and presents a Cartesian coordinate based display of the simulated network as shown in Figure 5.3. iNSpect provides the ability to see packets flow across the wireless network, as well as to collect various statistics. One feature of iNSpect that can easily be compared to NAM, is its ability to depict a broadcast. While NAM displays a circle as described earlier, iNSpect draws an arrow from the broadcasting node to all nodes within transmission range. Once a node is selected to forward the given packet, its color is changed to green to depict its win. All other nodes change their colors to black, showing that they are not the winners of that

election.

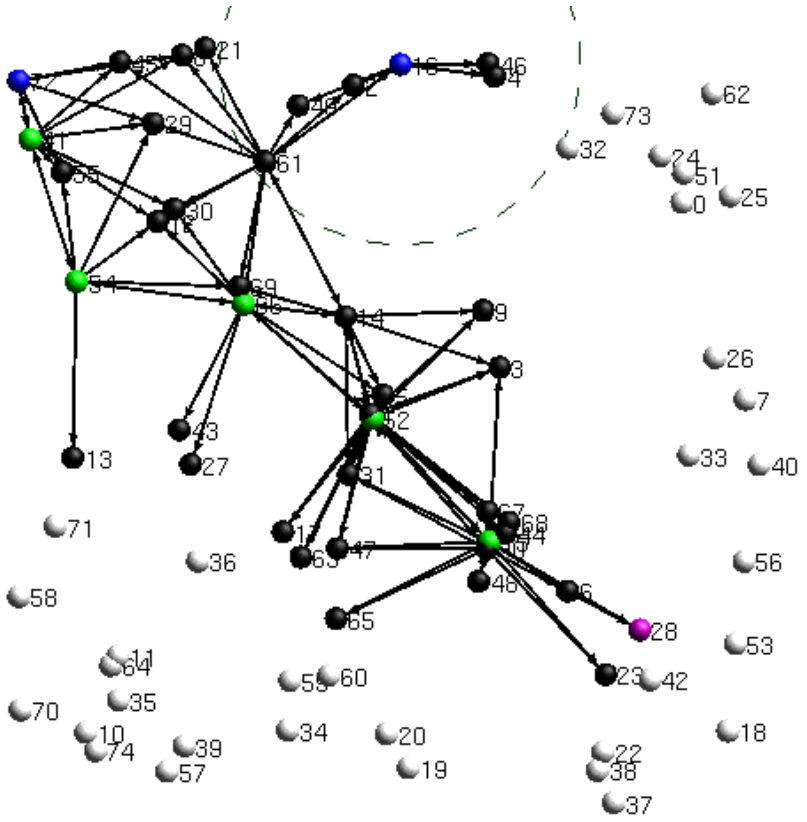

Figure 5.3: An example of the visualization window in iNSpect. In this thesis, black is used to identify nodes that intentionally drop packets, green are forwarding nodes, blue are sources, and purple are destinations.

Since SENSE was not originally created to produce ns2 type output, and iNSpect was created to use ns2 output, the main effort in making SENSE work well with iNSpect was to modify the output from a SENSE simulation to match the output of ns2. Once broken down to its most basic levels, this is a relatively simple process. iNSpect expects three separate input files to produce a visualization of a simulation. Those files are the configuration file, the mobility file, and the simulation file.

The configuration file is a user-created file based on a template provided with the iNSpect tool. The main items that must be modified in the configuration file are the physical dimensions of the simulated field, the number of nodes within the given

simulation, the start and end times of the portion of the simulation to be visualized, and the colors of the nodes. The color definition also includes the ability to separate events into different categories for statistic gathering. Within the configuration file, it is also possible to set the amount of simulation time that an event is visualized for before it reverts to its default state.

The mobility file is the file that iNSpect uses to present the physical locations of the nodes. In our work with SENSE and iNSpect, we've not provided the ability for nodes to move, so SENSE simply provides output that establishes the initial positions of the nodes. In SENSE, node locations are established at network setup, so a simple textual output in the format expected by iNSpect gave us all of the data required for the mobility file. This format is simply three lines for each node, which identify the X, Y, and Z coordinates of the given node.

Lastly, the simulation file is the most important part, and provides all of the information to the visualizer that concerns data transmissions. This includes sent packets, received packets, acknowledged packets and dropped packets. This is of course the most complex of the three files required by iNSpect. In order to provide the output required by iNSpect, the basic template provided in [33], which is also included here as Figure 5.4, was used. In order to create events for iNSpect to display, SENSE simply outputs a line of text that fills the fields presented. The fields include the ID or address of the node, the simulation time of the event to be displayed, a description of the event (either sending to or received from), the node on the other end of the event, or -1 for a broadcast, a status string, and the packet ID. The status string provides a great deal of flexibility in the use of the visualizer, as that string is used to provide both the node coloring in the simulation replay and the categories for statistic gathering. In our implementation, we use the status string to separate network initialization packets from data packets. In addition to this separation, the status string is used to identify source nodes, destination nodes, forwarding nodes, nodes that intentionally drop packets, and several other types of events.

As mentioned previously, the implementation of providing iNSpect-compatible output from SENSE, while effective, is also very simple. The extent of the work

Field and Description					
Node	Time	Event	Other Node	Status	Packet
ID	Seconds elapsed	*sending to* or *received from*	*ID* or *-1*	Custom string	ID

Figure 5.4: Requirements for an event in iNSpect.

required involves identifying when events in the protocol need to be displayed in the visualization. For our protocols, those events were the transmission of a packet, the reception of a packet, the winning of a forwarding election, and the intentional dropping of a packet due to the loss of an election. Each of these events simply required an output from the simulator into the simulation file that followed the format presented in the table in Figure 5.4. As protocols change, or different events within the simulator want to be focused on, a simple change of where text is output from the simulator is all that is required. It is also possible to change the iNSpect configuration file to no longer recognize specific status strings if the data represented by those status strings is not desired in the current visualization.

In addition to the basic visualization capabilities that iNSpect adds to SENSE, there are several other features that increase its value immensely. First is the ability to change the speed of the animation on the fly, including the ability to skip forward and backward in 5 second increments. If the only desired output is the statistics that are presented at the end of a simulation, the speed may be set to extremely fast. In addition, it is possible to rewind the animation, thus reviewing specific events within the simulation repeatedly. iNSpect also provides the ability to view node coordinates in the animation. Another highly useful tool within iNSpect is the ability to view a connectivity graph or conduct a partition check of the network. These tools give more insight into the physical layout of the nodes, and can help to identify connectivity issues within the network while troubleshooting a simulation. While iNSpect is running an animation, each time it changes colors due to an event, statistics are gathered to represent that event as well. At any time in the simulation, a user may print the currently collected statistics both for the network as a whole, or for individual nodes. Lastly, iNSpect provides the ability capture both still images

and animated movies for later replay. All of these features not only make iNSpect a great tool for providing wireless network visualization, but since it now functions well with SENSE, also provides excellent visualization capabilities for simulation results from a wireless sensor network specific simulator.

5.3 Using Visualization for Protocol Enhancement

Although visualization is a wonderful tool to have, it does not provide any added value if it is not put to good use. To that end, two specific examples of how SENSE with iNSpect has been useful in this work are described below.

5.3.1 Visualization to Modify DREQ/DREP

After enhancing SENSE with an interface to iNSpect, it was observed that the network establishment, or DREQ/DREP phase of the SRP protocol was not working correctly. As designed, each source node would only send a single DREQ packet for any given flow. It was observed that when implemented in SENSE, an error occurred that allowed many nodes to repeatedly send DREQ packets. This only occurred when the random time selected between packet transmissions at the source was less than the round trip time for the DREQ/DREP. However, when this time was small enough, a source would continually send DREQ packets until such time as the DREP returned. In some cases, this resulted in hundreds of DREQ packets being flooded across the network before the destination could respond with a DREP. Until visualization with iNSpect was implemented into SENSE, the only way to identify this problem would have been by searching line by line through simulator output to find repeated packets. With visualization, it was simply a matter of watching a DREQ/DREP cycle, and seeing that nodes were flashing with the same received DREQ color multiple times. Once the problem was identified, a simple fix was implemented, in which nodes which wanted to resend a DREQ were forced to wait 10 packet transmission cycles. This still resulted in a few nodes sending multiple DREQ packets, but cut the number of DREQ/DREP packets drastically.

SENSE and iNSpect were also used to validate this modification to ensure that the protocol was now functioning as desired. In order to validate this change, the

statistic printing function that was mentioned earlier was used. If the protocol is working at its absolutely optimum during DREQ/DREP, each node will see $n-1$ DREQ packets and $n-1$ DREP packets, where n is the total number of nodes in the network. The source that originates the DREQ and the destination the originates the DREP will not receive the packets they created, hence $n-1$. This optimum is due to the fact that both the DREQ and DREP are flooded across the entire network, and ideally, each source only sends one DREQ and each destination only sends one DREP. Before the modification to the protocol, the total number of DREQ packets was approximately $3n$, while the number of DREP packets was approximately $1.5n$, depending on the random seed used. After modification, the total number of DREQ packets was within $n/10$ of n and the number of DREP packets was exactly n for every seed tested.

5.3.2 Visualization to Create Variable λ

In addition to modifying the DREQ/DREP stage of SRP, iNSpect was also used to validate a modification to the selection of λ that the protocol uses. In the original design of all of the SSR protocols, λ was a fixed value that was selected at run time. Generally, we've maintained $\lambda = 100$ms. This was a fine solution, except for the fact that individual nodes have different numbers of neighbors. Those nodes with more neighbors may require a random delay range larger than 0 to 100ms to avoid collision, while those nodes with few neighbors may allow a significantly smaller range to be used. Initially, iNSpect was used to get a general idea of the high and low values of numbers of neighbors. Figure 5.5 is an iNSpect generated plot with the connectivity graph displayed on a randomly generated network of 75 nodes in a 1000m by 1000m space with a 230m transmission range. It is easily observed that there is a large variation in the number of neighbors that nodes have. Some nodes, such as node 26, have as few as 5 neighbors, while other nodes, such as node 47, have as many as 17 neighbors. This large variation in density results in a protocol that is not completely efficient with a fixed value of λ. In the example of node 47, 17 neighbors means that on average, there will only be a 5.8ms gap between delay selections, which could lead to a higher probability of collision. On

the other hand, node 26 has only 5 neighbors, which means there will be an average 20ms gap between delay selections, resulting in a low probability of collision, but a fairly inefficient node. Variable λ corrects this variation.

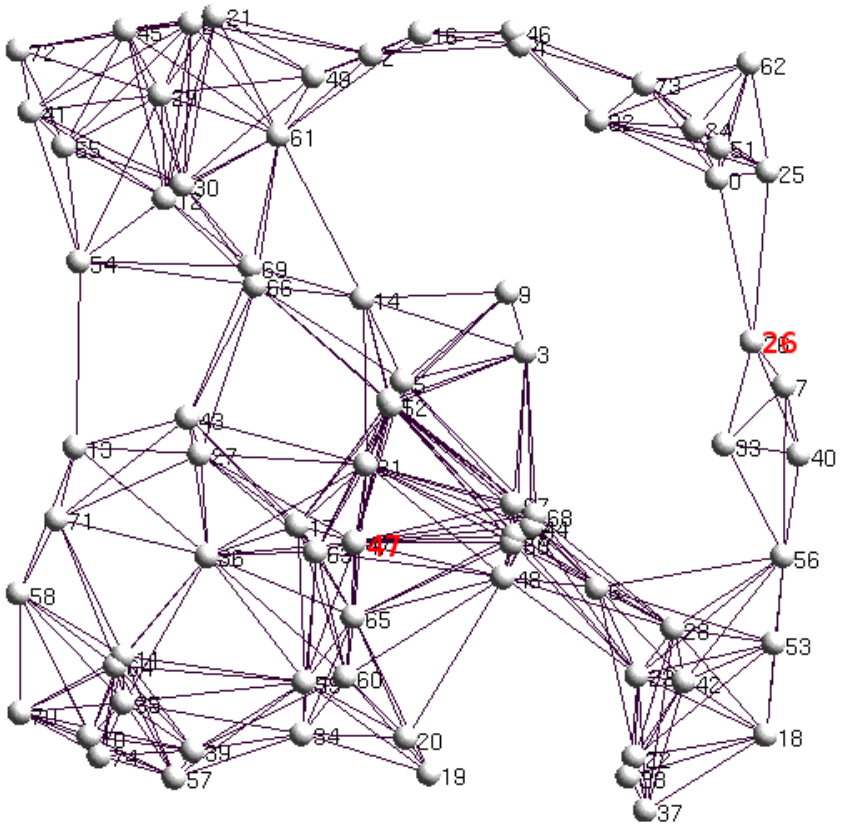

Figure 5.5: An example of the connectivity graph provided in iNSpect that was used to identify the variation in the number of neighbors throughout the network.

After validating that the number of neighbors varied as much as it does, a simple solution was devised in which each node calculates how many neighbors it has by using the first DREQ sent across the network. Since every node will only forward a DREQ packet once, all that is required is to have each node count the number of times that it receives that packet. Upon completion of the initial DREQ packet's transmission, each node multiplies the number of neighbors by 15ms to generate its value for λ. Since the majority of networks that we work in have an average number of neighbors between 7 and 8, this puts the average value of λ between 105ms and 140ms. Although this value is greater than the original 100ms,

because of the reduced number of collisions, it actually results in faster throughput with higher reliability.

While the varying number of neighbors is easily identifiable using iNSpect's Connectivity Graph feature, with SENSE alone, it would require a fair amount of output deciphering and math to calculate how many nodes are within the transmission range of each other. Figure 5.6 shows a sample of SENSE output that gives location information. Each line represents a different node at simulation time 0.0, and provides the node's identity, in addition to each node's location in the simulation field.

```
@ 0.000000 48 619.448833 312.245978
@ 0.000000 47 439.345178 354.177704
@ 0.000000 46 631.524183 973.099584
@ 0.000000 59 378.585883 185.356447
@ 0.000000 45 161.746116 975.218554
@ 0.000000 44 653.428245 373.588012
@ 0.000000 43 237.211018 505.713423
@ 0.000000 42 838.090571 186.454483
@ 0.000000 41 46.932318 876.321646
@ 0.000000 40 977.632872 461.077431
@ 0.000000 39 243.668783 102.257224
@ 0.000000 60 428.248296 191.983085
@ 0.000000 38 770.983921 71.046053
@ 0.000000 37 791.883892 29.958732
@ 0.000000 36 260.913117 338.199724
@ 0.000000 35 159.460456 160.431920
```

Figure 5.6: An example of the location output from SENSE.

6. Discussion and Conclusions

This thesis has presented SRP, which leverages the fault tolerance of dynamic path selection in WSNs with the speed of a static path routing. Under SRP, the intermediate nodes on routes to the destination use broadcast communication and prioritized transmission back-off delay to make packet forwarding decisions based only on their hop distance from the destination and their previous forwarding decisions. The sender of the packet listens to the responses, and if there is no response to the original and repeated transmissions (a clear sign of faults in its neighborhood), it increases its hop distance to the destination to facilitate seamless route repair. Additionally, SRP improves the end-to-end delay of the protocol by allowing the node that has won the self-selection to cut its back-off delay to near zero for subsequent packets of the same flow. This node then is assured of winning the subsequent self-selections until failure of the node or its link removes it from the competition (in which case its back-off delay is restored to the original value). This design creates a protocol that is fast when there are no failures and yet robust when failures do occur.

In addition to protocol enhancements, the combination of SENSE and iNSpect has provided the ability to convert pages of simulation output, into a clear and concise view of our networks. The animations presented through this combination have been invaluable in the continued development of the family of Self-Selective Routing protocols. They allow the presentation of textual simulator output in such a manner that problems are easily identifiable, while still maintaining the basic ability to gather statistics and keep things simple. The ability to find a visualization tool that would work easily with the simulator allows the research focus to remain on our simulator and protocols, while maintaining access to the state of the art in wireless network simulator visualization tools. The combination of these tools will continue to be very useful to SENSE users all over the world in support of their research on wireless sensor networks.

Several possibilities exist for extending this research. If the desire is to pur-

sue the path of protocol improvement, researchers could focus on improving the robustness of SRP by adding support for energy management and mobility. Currently, SRP is much more energy efficient than any of the previous SSR protocols, and is certainly more efficient than AODV, but a simple node sleep schedule could be implemented to improve that efficiency even more. Additionally, SRP currently provides no support for mobile nodes. A simple idea that certainly could be improved upon would be to set a regular schedule for a network discovery to take place. This would re-establish node locations in their respective flows, allowing the rest of the protocol to remain the same. Of course this would be inefficient, but is a starting point for mobility research. If the desire is to pursue the path of visualization improvement, researchers could work to more tightly connect iNSpect and SENSE, with the possibility of visualizing real-time simulations. iNSpect is built to handle real-time input from ns2, so the extension to providing that capability within SENSE should not be too large a step. In addition to this, one could add the capability of visualizing mobile nodes or node power levels in conjunction with researchers adding those capabilities to the SSR protocols. iNSpect is built to handle mobile nodes, and would simply require updates to the mobility file. The addition of energy level visualization would simply be an additional output into the simulation file that would be tracked as an additional event to those already taking place.

LITERATURE CITED

[1] A. Woo, T. Tong, D. Culler: Taming the underlying challenges of reliable multihop routing in sensor networks. Proc. ACM SenSys03, ACM Press, New York, 2003, pp. 14-27.

[2] J. Zhao, R. Govindan: Understanding packet delivery performance in dense wireless sensor networks. Proc. ACM SenSys 03, ACM Press, New York, 2003, pp. 113.

[3] G. Anastasi, A. Falchi, A. Passarella, M. Conti, E. Gregori: Performance measurements of motes sensor networks. Proc. 7th ACM Intern. Symp. Modeling, Analysis and Simulation of Wireless and Mobile Systems, ACM Press, New York, 2004, 174-181.

[4] Crossbow Technology, Inc., http://www.xbow.com

[5] C. Perkins, E. Belding-Royer, S. Das: RFC 3561-ad hoc on-demand distance vector(AODV) routing. http://www.faqs.org/rfcs/rfc3561.html

[6] C. Intanagonwiwat, R. Govindan, D. Estrin: Directed diffusion: a scalable and robust communication paradigm for sensor networks. Proc. ACM MobiCom, ACM Press, New York, 2000, pp. 56-67.

[7] J.W. Branch, M. Lisee, B.K. Szymanski: SHR: Self-Healing Routing for wireless ad hoc sensor networks. Proc. Intern. Symp. Performance Evaluation of Computer and Telecommunication Systems SPECTS'07, SCS Press, San Diego, 2007, pp. 5-14.

[8] B.K. Szymanski, C. Morrell, S.C. Geyik, T. Babbitt: Biologically Inspired Self Selective Routing with Preferred Path Selection. Bio-Inspired Computing and Communication, LNCS, vol. 5151, Springer, New York, NY, 2008, pp. 217-228.

[9] R. Poor: Gradient routing in ad hoc networks. http://www.media.mit.edu/pia/Research/ESP/texts/poorieeepaper.pdf

[10] F. Ye, G. Zhong, S. Lu, L. Zhang: Gradient broadcast: a robust data delivery protocol for large scale sensor networks. ACM Wireless Networks, 11(2) (2005).

[11] M. Heissenbttel, T. Braun, T. Bernoulli, M. Waelchli: BLR: beaconless routing algorithm for mobile ad hoc networks. Computer Communications Journal, 27(11)(2004).

[12] M. Zori, R.R. Rao: Geographic Random Forwarding (GeRaF) for ad hoc and sensor networks: multihop performance. IEEE Trans. Mobile Computing, 2(4) (2003) 337-348.

[13] B. M. Blum, T. He, S. Son, J.A. Stankovic: IGF: a robust state-free communication protocol for sensor networks. Technical Report CS-2003-11, University of Virginia, Charlottesville, 2003.

[14] Y. Xu, W.C. Lee, J. Xu, G. Mitchell: PSGR: priority-based stateless geo-routing in wireless sensor networks. Proc. IEEE Conf. Mobile Ad-hoc and Sensor Systems, IEEE Computer Society Press, Los Alamitos, 2005.

[15] D. Chen, J. Deng, P.K. Varshney: A state-free data delivery protocol for multihop wireless sensor networks. Proc. IEEE Wireless Communications and Networking Conf., IEEE Computer Society Press, Los Alamitos, 2005.

[16] C. Johnson: Visualization viewpoints. IEEE Computer Graphics and Applications (2004), 13-17.

[17] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Halmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, H. Yu: Advances in network simulation. IEEE Computer 33(4) (2000), 59-67.

[18] C. Goldstein, S. Leisten, K. Stark, A. Tickle: Using a network simulation tool to engage students in active learning enhances their understanding of complex data communications concepts. 7th Australasian conference on Computing education, Australian Computer Society, Darlinghurst, Australia, 2005, pp. 223-228.

[19] K. Fall, K. Varadhan (eds.): The ns manual (formerly ns notes and documentation). The VINT Project, 2008, http://nsnam.isi.edu/nsnam/index.php.

[20] S. Kurkowski, T. Camp, N. Mushell, M. Colagrosso: A visualization and analysis tool for ns-2 wireless simulations: inspect. Proc. of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS) (2005), 503-506.

[21] G. Chen, J. Branch, M.J. Pflug, L. Zhu, B. Szymanski: Sense: A sensor network simulator. Advances in Pervasive Computing and Networking (2004), 249-267.

[22] G. Chen, B.K. Szymanski: Cost: A component-oriented discrete event simulator. Proc. Winter Simulation Conference, WSC02 (San Diego, CA), vol. I, December 2002, pp. 776-780.

[23] B.K. Szymanski, G.G. Chen: Sensor network component based simulator. Handbook of Dynamic System Modeling (Paul Fishwick, ed.), CRC/Taylor and Francis Publishing, 2007, pp. 35-1 – 35-16.

[24] H.L. Xuan, S. Lee: Two energy-efficient routing algorithms for wireless sensor networks. Networking, LNCS, Springer, New York, NY, 2005, pp. 698-705.

[25] H.K. Ryu, Y.Z. Cho, D.H. Kim, K.W. Lee, H.D. Park: Improved handoff scheme for supporting network mobility in nested mobile networks. Computational Science and Its Applications, LNCS, Springer, New York, NY, 2005, pp. 344-347.

[26] T.T. Huynh, C.S. Hong: A novel hierarchical routing protocol for wireless sensor networks. Mobile Communications Workshop, LNCS, Springer, New York, NY, 2005, pp. 339-347.

[27] G. Chen, J. Branch, B.K. Szymanski: Local leader election, signal strength aware flooding, and routeless routing. 5th IEEE Intern. Workshop Algorithms for Wireless, Mobile, Ad-Hoc Networks and Sensor Networks WMAN 2005. IEEE Computer Society Press, Los Alamitos (2005).

[28] S. Koenig, B.K. Szymanski, Y. Liu: Efficient and Inefficient Ant Coverage Methods. Annals of Mathematics and Artificial Intelligence 31(1-4), 4176 (2001).

[29] T.S. Rappaport: Wireless Communications: Principles and Practice. Prentice Hall, Englewood Cliffs (1996).

[30] K. Wasilewski, J. Branch, M. Lisee, B.K. Szymanski: Self-healing routing: a study in efficiency and resiliency of data delivery in wireless sensor networks. Proc. Conference on unattended Ground, Sea, and Air Sensor Technologies and Applications, SPIE Symposium on Defense & Security, April, Orlando, FL (2007).

[31] J. Glaser, D. Weber, S.A. Madani, S. Mahlknecht: Power aware simulation framework for wireless sensor networks and nodes. EURASIP Journal on Embedded Systems 2008 (2008).

[32] D. Estrin, M. Handley, J. Heidemann, S. Mccanne, Y. Xu, H. Yu: Network visualization with the VINT network animator nam. Technical Report 99-703, University of Southern California, 1999.

[33] S. Kurkowski, T. Camp, M. Colagrosso: A visualization and analysis tool for wireless simulations: inspect. ACM's Mobile Computing and Communications Review, to appear (2008).

[34] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister: System architecture directions for networked sensors. Proc. 9th ACM Int. Conf. Architectural Support for Programming Languages and Operating Systems, 2000, pp. 93104.

[35] T.T. Huynh, C.S. Hong: An energy delay efficient multi-hop routing scheme for wireless sensor networks. IEICE Transactions on Information and Systems E89(D5) (2006) 6541661.

[36] T. Babbitt, C. Morrell, B.K. Szymanski, J. Branch: Self-Selecting Reliable Path for Wireless Sensor Network Routing. Computer Communication Journal, vol. 31, no. 16, 2008, pp. 3799-3809.

[37] C. Morrell, T. Babbitt, B.K. Szymanski: Visualization in Sensor Network Simulator, SENSE and Its Use in Protocol Verification. Technical Report 08-13, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, 2008.

[38] B.K. Szymanski, G. Chen: Computing with Time: From Neural Networks to Sensor Networks. The Computer Journal, vol. 51(4):511-522, 2008.