WIRELESS SENSOR NETWORK

SIMULATOR

Graduate Project

by

ADRIAN SALAZAR

Submitted to the Office of Graduate Studies of
Texas A&M University-Corpus Christi
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

MAY 2010
ABSTRACT

Wireless Sensor Networks (WSNs) are composed of commodity nodes that are able to sense, compute data and communicate with each other. Nodes have the capability of being deployed both statically and dynamically by being placed at strategic locations or being dropped from an airplane. In addition, the nodes are able to self configure and form a network. Through the use of various routing protocols, the nodes are capable of communicating with each other. While discovery and routing techniques can be complex, through the use of a simulator both novice and advanced researchers are able to understand and explore the various protocols.

By surveying different methods of node configuration, routing models and protocol a WSN simulator can be successfully built to deploy a virtual WSN. Using different features such as topology control, low level network simulation, and various node and network constraints implemented by simulators like TOSSIM, Atarraya and NS-2 the simulator is able to accurately represent a working WSN. The simulator provides researchers feedback on the network that can be used to analyze different routing and discovery protocols. Although the simulator focuses on select discovery and routing protocols, it provides a foundation for future work and expansion into this topic.
DEDICATION

This work is dedicated to my mother Bertha Salazar and to the memory of my father, Greg N. Salazar, without whose help and encouragement none of this would be possible.
ACKNOWLEDGEMENTS

A special thanks to the faculty of Texas A&M University-Corpus Christi who served on my committee, Dr. Michael Scherger (chair), Dr. Scott King, and Dr. Ahmed Mahdy. These professors served as a backbone of guidance, encouragement and friendship not only in the process of this work but through my career here at this fine institution. A special recognition to Dr. John Fernandez, chair of the Department of Computing Science, who provided me with countless opportunities within the research community and encouraged a love for research.

To my family, a special thanks for supporting me and sacrificing so much so that I may fulfill my dreams. To my brother and sister Joel and Bennie Martinez, I love you all so much and appreciate all the work that you have all done to put me through school.

Finally, my gratitude to Cory D. Mathews. An alumni of the university who helped in the testing and debugging process. Without his help, the end result would be very different.
# TABLE OF CONTENTS

ABSTRACT .............................................................................................................. i
DEDICATION .......................................................................................................... ii
ACKNOWLEDGEMENTS ...................................................................................... iii
TABLE OF CONTENTS .......................................................................................... iv
LIST OF FIGURES ................................................................................................... vi

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>WIRELESS SENSOR NETWORKS</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>WIRELESS SENSOR NETWORKS</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Initial Node Configuration</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Routing Strategies</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Routing Protocols</td>
<td>9</td>
</tr>
<tr>
<td>II</td>
<td>EXISTING WSN SIMULATORS</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>TOSSIM</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>NS</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>Atarraya</td>
<td>20</td>
</tr>
<tr>
<td>III</td>
<td>WSN SIMULATOR REQUIREMENTS</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Supported Protocols</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Node Constraints</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>User Interface</td>
<td>25</td>
</tr>
<tr>
<td>IV</td>
<td>WSN SIMULATOR DESIGN</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Node Attributes</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Node Deployment</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Node Discovery</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Flooding Protocol</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>Gossiping Protocol</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>Power Production and</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>Consumption</td>
<td></td>
</tr>
</tbody>
</table>

4
<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>SIMULATION RESULTS AND ANALYSIS</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>Node Deployment</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>Node Discovery</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>Flooding Protocol</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>Gossiping Protocol</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>Power Consumption</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>41</td>
</tr>
<tr>
<td>VI</td>
<td>CONCLUSION AND FUTURE WORK</td>
<td>43</td>
</tr>
<tr>
<td>VII</td>
<td>REFERENCES</td>
<td>46</td>
</tr>
<tr>
<td>FIGURE</td>
<td>Figure 1</td>
<td>Basic Hello Protocol</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Energy Aware Hello Protocol</td>
<td>4</td>
</tr>
<tr>
<td>Figure 3</td>
<td>One Hop Routing Strategy</td>
<td>6</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Multi Hop Routing Strategy</td>
<td>7</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Cluster-based Hierarchical Routing Strategy</td>
<td>8</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Gossiping Example</td>
<td>11</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Ideal Dissemination</td>
<td>13</td>
</tr>
<tr>
<td>Figure 8</td>
<td>SPIN-1 Protocol</td>
<td>14</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Random Node Deployment Algorithm</td>
<td>28</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Static Deployment Algorithm</td>
<td>29</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Node Deployment Algorithm</td>
<td>30</td>
</tr>
<tr>
<td>Figure 12</td>
<td>von Neumann Node Neighborhood</td>
<td>30</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Distance Formula Algorithm</td>
<td>31</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Pythagoreans Theorem</td>
<td>31</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Classic Flooding Algorithm</td>
<td>32</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Gossip Algorithm</td>
<td>33</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Node Deployment Using Various Number of Nodes</td>
<td>37</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Node Discovery Using Various Number of Neighbors</td>
<td>38</td>
</tr>
<tr>
<td>FIGURE</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>Figure 19  Flooding Protocol Example</td>
<td>39</td>
<td></td>
</tr>
<tr>
<td>Figure 20  Gossip Protocol Example</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>Figure 21  Power Consumption by Protocol</td>
<td>42</td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER I
WIRELESS SENSOR NETWORKS

Wireless Sensor Networks (WSNs) are composed of commodity nodes equipped with a sensor, transceiver, micro controller as well as a power source. These networks are application specific and can range in size of up to several thousand nodes. Deployment can be done statically by placing them at strategic locations or dynamically using random deployment methods such as being scattered from an airplane over treacherous, hostile or otherwise unreachable terrain.

The use of WSN has begun to increase as of recent years. The ability to self configure and their resilience have made them popular choices for tasks ranging from area monitoring to security and intrusion detection in both civil and military fields.

Once deployed, nodes have the capability to self configure using a discovery protocol and form neighborhoods. Node communication from sink or base station to each node is determined by the routing strategy in place. Reliable routing is crucial to the success of WSNs where up to the minute data is required for many applications. While there has been a significant amount of research into this topic, determining the best protocols to be used in any situation require testing of the environment that a node will be in. Testing on the actual environment however is not a feasible option due to monetary and time constraints.

This report has been broken down into different chapters, the remainder of Chapter I focuses on the structure of a WSN and surveys the various discovery and
routing techniques and protocols. Chapter II surveys and analyses the existing simulators, while chapter III discusses design issues of the new visual simulator. Chapter IV and V analyze the new simulator and the results are presented followed by the conclusion and future work in Chapter VI.

1.1 INITIAL NODE CONFIGURATION

Wireless Sensor Networks (WSN) are very uncharacteristic of other networks. Initial configuration of a network is not done manually like is the case with current computer networks; WSN nodes are capable of self-configuring in order to form a reliable network. Unlike other types of networks, a node does not need to connect to a router directly, it can use other nodes to connect and pass messages while interacting with the environment; this is typical of ad-hoc networks. When a node is initially deployed, the first step that is performed to configure the network is to determine each node’s neighboring node and identify a possible link through the use of a connection or neighborhood table. The table serves as a means of not only routing messages but also aids in maintaining the network.

Node discovery is not limited to the initial node deployment; discovery can also take place when new nodes join an existing network or when node failure occurs. Discovery can take place through various Hello Protocols in which a node transmits a “Hello” packet and listens for a reply. If an acknowledgement is received a node will then add the acknowledging node to its connection table.
One of the more prominent, and simplistic protocols that are currently being used is that of the Basic Hello Protocol. In this protocol, a node can be in one of two states; talking or listening. The two states occur within a time frame \( w \) of duration \( f \), as illustrated in Figure 1.

![Figure 1. Basic Hello Protocol](image)

In each time frame, a node will randomly choose a starting time to transmit a “Hello” message for a predetermined amount of time. The packet transmission occurs with no carrier sensing or checking to see that the channel in which it is sending the packet is clear. A node can only transmit a single “Hello” packet per time frame; if collision occurs during the current time frame it must wait till the next time frame before it can resend a packet [1].

Given the information, an equation can be derived to represent the medium access probability of \( \frac{T}{w} \) where \( T \) is the length of the talking state. The probability of successfully sending a packet without collisions is equal to Equation 1,

\[
\left(1 - \frac{2T}{w-T}\right)^{n-1}
\]  

Equation 1.
where \( n \) is the number of nodes within the collision range [1]. The Basic Hello Protocol however suffers from one major flaw; power consumption from being idle. In order to maintain the life of a WSN each node must use power conservatively.

A different version of the Basic Hello protocol is that of the Energy Aware Hello Protocol which incorporates three different states; talking, listening and sleeping. As seen in Figure 2, the protocol addresses the power constraint by entering a sleep state during periods of inactivity. During this time the node will turn the radio interface off and is unable to communicate with the other nodes.

![Figure 2. Energy Aware Hello Protocol](image)

At each timeframe a node will pick an initial time to wake up and begin listening, it will then pick another time to initiate packet transmission. While this protocol is wise to conserve power and reduce collisions it also introduces a new problem; discovery of neighbors that may be sleeping. The probability of node discovery for this protocol is demonstrated in equation 2,

\[
\frac{w-T}{w+S} \left( 1 - \frac{2T}{w+S} \right)^{n-2}
\]

Equation 2.
where $s$ is the sleeping time and $n$ is the number of nodes within the vicinity. Trade-offs are common in WSN, however due to periodic maintenance nodes that might of not been discovered initially will join the network as time progresses [1].

Both Basic and Energy Aware Hello Protocols represent two different ways of successfully achieving node discovery. Both protocols have tradeoffs and depending on the purpose and intended duration of a WSN one protocol might prove better use over the other.

1.2 ROUTING STRATEGIES

Routing within a WSN is very different from routing in traditional computer network. WSN are dynamic in nature, they can be in various stages at any given time and may not always have the same neighbors or be in the same location. Existing strategies such as TCP/IP routing cannot be employed, rather new routing strategies are implemented.

1.2.1 One Hop Routing

Perhaps one of the simplest way to achieve routing is to use a centralized node to route messages through the network as seen in Figure 3. This strategy, known as One-Hop Routing is simple to implement, however it is only feasible under certain conditions such as a small contained WSN.
Figure 3: One Hop Routing Strategy

Nodes in a WSN are typically scattered over a wide area and are limited not only by power constraints but by their transceivers transmission range. Although nodes may capable of forming such type of network, the nodes will be hindered by the power consumption that is required to transmit messages over a long distance. This results in a shorten lifespan of the network. Overall, this type of model is not a good fit for most WSN applications. [2].

1.2.2 Multi Hop Routing

A different type of routing strategy that enjoys success with WSN is that of multi-hop routing. Under this model a node will forward a packet to one of its neighbors and in turn the neighbor will forward the packet to its neighbors as seen in Figure 4. The process is repeated until the sink or a base station is reached [2].
However due to the limitations of a WSN routing protocols that previously worked on interconnected networks and Mobile Ad-Hoc Networks (MANETS) that use a multi-hop model are no longer reliable. Internet routing requires reliable connections and addressing while MANETS may assume that certain nodes will always be available for routing a message, two characteristics that a WSN does not possess. Routes in a WSN are highly dynamic. Nodes are not always in operation; they can switch between active and inactive states or might even suffer from node failure and mobility [3].

There are two different approaches when it comes to multi-hop routing; node centric and data centric. Node centric routing uses an addressing scheme that requires the knowledge of the node that initiated the packet. Data centric routing however is interested in the data and the area where the packet originated from [4]. Various multi-hop routing protocols are able to achieve such routing schemes.
1.2.3 Cluster Based Routing

One of the goals of any routing strategy is to make a network as useful and efficient as possible. Under a cluster based routing strategy, different nodes form groups in order to efficiently relay data to the sink or base station. For every cluster there is a cluster head. These nodes perform aggregation and are able to send a coherent packet that reflects the data sensed by the cluster to the base station [8]. Figure 5 demonstrates such type of network, where the nodes join a cluster head to form a sub network using a one-hop routing. The cluster heads in turn form a separate network that is able to communicate with the base station through multi-hop routing.

![Figure 5. Cluster-based Hierarchical Routing Strategy](image)

In addition of having normal nodes being grouped together to form a cluster, cluster heads can also be grouped together to produce a cluster of cluster-heads. As a result layering is being introduced to the network producing a hierarchical model. Data is able to travel from a lower cluster layers to a higher one, as a result data is able to
travel quickly through the network. Cluster based routing implements both one-hop routing as well as multi-hop routing. Nodes are connected directly to a cluster node. When a packet is passed to the cluster head, the cluster head will forward it to other cluster heads performing a series of hops. Unlike the traditional multi-hop routing model, cluster based multi-hop is able to traverse the network faster over larger distances while reducing the latency and performing data aggregation [8].

1.3 ROUTING PROTOCOLS

Traditional routing protocols, much like the routing strategies are no longer valid options for WSNs. There are no longer address-based destinations, rather a more general approach such as geographic location, information gain and sink locations are used as routing metrics [5]. In addition to the lack of addresses in WSN, other metrics such as node energy, state and link reliability are considered when developing a routing protocol.

1.3.1 Flooding

Flooding is one of the most trivial routing protocols used in WSN. It utilizes a multi-hop routing schemes. It is a highly inefficient but yields positive results in highly mobile WSN. Its use is limited to node and route discovery as well as maintaining existing routes [5].

Flooding typically takes place from the sink node to the various nodes. Basic flooding occurs when the sink node transmits a control packet to all of its neighbors. The neighbors then take the packet and send the message to their neighbors. While this is one
way to ensure that a packet is delivered, it also makes unnecessary packet transmissions. Nodes will ultimately have redundant copies of data and the probability of packet collision increases.

A viable flooding protocol that is used to determine shortest path routes from nodes to sink is that of Constrained Flooding. Under this protocol, cost-to-go values are maintained by each node, where the cost-to-go value is represented by the number of hops. Initially the sink node will have a hop value of zero while the rest of the nodes have an infinite value. The sink node sends its cost-to-go to its neighbors. The neighbors do the same while maintaining the lowest cost-to-go value. These steps are repeated till all nodes have received a value. In an effort to avoid packet collisions, constrained flooding uses random delays between transmissions. However it still suffers from a high communication overhead. The cost-to-go value will remain true while the network remains static. If a node failure or any other changes to the network occur then the process is repeated. This incurs extra overhead for the WSN [5].

1.3.2 Gossiping

A gossip protocol takes a more conservative approach to routing as opposed to the flooding protocol. Under gossiping a node will randomly choose a node to forward a packet. Figure 6 demonstrates a gossiping example where node A is attempting to send to node C. Initially, A will select a neighbor to send the packet. When node B receives the packet it will choose a random neighbor to forward the packet to, node D, whose only neighbor is B. When B receives the packet back from D, it will select a neighbor
who it has not yet communicated with. Node B will then send the packet to C who is the intended receiver and the routing would terminate.

![Figure 6. Gossiping Example](image)

Gossiping can be implemented using a sender and receiver approach [6, 7]. Sender approach gossiping closely follows an epidemic spread where one node that is infected can contaminate a random set of nodes; it can forward them the packet. This however requires that the node know ahead of time if the node is in a state to receive the packet.

Similarly, under receiver based gossiping whenever a node receives a packet it can choose whether it wants to forward the packet or not. The node will randomly select a subset of its neighbors that it wishes to forward the packet to.

Received-based gossiping is used as a broadcasting protocol to disseminate data through the network. It uses a probabilistic approach to forwarding the packets.
discard the packet $1 - p$. Duplicate packets are ignored under this topology. Gossip1 protocol uses this scheme although other versions of this protocol have been proposed [6].

Another proposed gossiping protocol that seeks to improve on the probability of successfully forwarding a packet is that of *Gossiping using the Energy Map* (GEM), a gossiping protocol that performs energy-aware broadcasting as opposed to random selection [6]. Performing energy aware broadcasting will not only ensure that the packets are successfully forwarded by avoiding low power obstacles but also prolong the life of the network by reducing the energy consumption. An energy map is formed by polling the nodes about their remaining power at each part of the network. GEM is a robust alternative to classic flooding; it has proven to be independent of the number of obstacles.

1.3.3 SPIN

Sensor Protocols for Information via Negotiation, SPIN, is a family of adaptive protocols that efficiently disseminates information among sensors in an energy constrained WSN [7]. These protocols use high-level data descriptors called meta-data to eliminate redundant packets from being passed around a network.

To obtain ideal dissemination all nodes send observed data along a shortest path route. Every node can then receive distinct data without receiving any duplicates in the shortest amount of time. Such example of ideal dissemination is seen in Figure 7 where events $a$ and $c$ occur. Initially, A and C will broadcast the event to their neighbors. Node
B is not aware of a and c so it will accept the packets. When A broadcast to C, node C already has packet c so it will only acknowledge packet a. C is unaware of event a so it only broadcasts packet c. In the second step, nodes A, B and C are aware of events a and b, node D is only aware of c. When node B broadcast the copy of the packets, D only accepts packet a. Through the ideal dissemination process, it ensures that energy use gets maximized by limiting unnecessary sending and receiving.

![Figure 7. Ideal Dissemination](image)

The SPIN family of protocols has multiple protocols that yield low power dissemination, SPIN-1 and SPIN-2. SPIN-1 uses simple handshaking to disseminate data throughout the network. It is a three stage protocol that uses ADV, REQ and DATA states. In the ADV state, a node that has information that it wants to disseminate will advertise the data by sending an ADV message to its neighbors. When a node receives the ADV message, it will check to see whether it already has the data. If the receiving node concludes that the message is new and wants to receive it then it sends back a REQ message. Upon receiving the REQ message, the node will begin to transmit the DATA back to node that made the request. Nodes are not forced to send back request, they have
the option to receive or ignore. Furthermore if a node wishes to aggregate their own data to the message they can do so, the node simply adds the data before initiating an ADV message [7].

Figure 8 demonstrates such process, initially (a) a node will send out an ADV packet to its neighbors who will in turn send back a REQ packets (b), once the packet has been received the DATA will be sent to all acknowledging nodes (c). To further disseminate the data through the network, a new ADV message is broadcasted (d) and a REQ (e) is received from interested nodes. The DATA will then be forwarded to the select neighbors (f). This process will continue till no more REQ messages are received.

![Figure 8. SPIN-1 Protocol](image-url)
Another protocol that also yields low power dissemination is that of the SPIN-2 protocol. This SPIN protocol makes a modification to the way that SPIN-1 functions. It adds a low-energy threshold to dictate whether a node participates in the routing process. When a node's energy level is above the threshold, a node will engage in the handshake process to disseminate data. However, if the energy levels were to drop below the threshold then it uses a conservative approach to see if it can successfully send and receive a packet. During this time if a node receives an ADV packet, it will only respond if it can successfully send out the REQ packet and receive the DATA. To conserve energy, if the node is below the threshold it will not be allowed to initiate the handshake process [7].

1.3.4 LEACH Protocol

One of the earlier protocols that make use of hierarchical clustering is that of Low Energy Adaptive Clustering Hierarchy, LEACH. This protocol selects a few sensors as cluster-heads and rotates the role to evenly distribute the energy load. LEACH does periodic centralized data communication within the cluster. Data is aggregated and compressed before being forwarded towards the base station.

LEACH is broken down into two phases; setup and steady phase. During the setup phase, cluster-heads are elected. Cluster-head selection is based on a predetermined fraction of nodes that will become cluster-heads, $p$, randomness, $r$ and a threshold, $T(n)$, that is calculated based on the following:

$$T(n) = \frac{p}{1-p(r \mod \left(\frac{1}{p}\right))} \text{ if } n \in G$$

Equation 3.
where $G$ is the set of nodes that are involved in the cluster-head selection process [9]. When a cluster-head is selected it will broadcast its status and based on the cluster-heads broadcasting strength, nodes that are within transmitting range will join the cluster-head to form a cluster.

During the steady state phase, nodes are able to transmit data to the cluster-head. The cluster-head will receive the data and aggregate it and compress it before sending it off once more. After a period of time the network will enter the setup phase once more and select a new cluster-head [9].

1.3.5 TEEN

Another low power hierarchical routing protocol that overcomes LEACH’s inability to provide time critical results due to periodical messaging is that of TEEN, Adaptive Periodic Threshold-sensitive Energy Efficient sensor Network Protocol. Designed for time critical applications, TEEN monitors the media listening for any incoming messages but it limits the amount of communication. The protocol utilizes a soft and hard threshold values to limit communication and increase the life of the network [9].

A sink node will initially send out a soft threshold value to the different cluster heads. The cluster heads in return will create a hard threshold value and send them to the cluster nodes. The nodes sense their environment continuously; if a parameter ever reaches the hard threshold value then it will turn on its transmitter and send the data to the cluster-head. The cluster-head will determine if the parameter is equal to or greater
than the soft threshold value. In doing so, only time critical data is being sent making the
network as efficient as possible when dealing with communication and power
consumption. Perhaps one of the biggest flaws that TEEN suffers from is if a threshold
value is never received. When this occurs the nodes will retain the old threshold value
and will never communicate leaving the user without any data from the network [9].
Presently there are various network simulators designed to simulate wireless sensor networks. Most simulators however do not model the complete network itself, rather they focus on individual components and attributes such as the operating system, network topology and routing. Some of the more prominent simulators used in the research field include TOSSIM which is a part of TinyOS, Atarraya and NS-2. Through the process of reviewing such simulators, a better simulator may be capable of being developed that can incorporate the different simulator strengths.

2.1 TOSSIM

TOSSIM is a discrete event simulator for TinyOS network [12, 13]. While the simulator is capable of simulating a WSN, it is not intended to be used as an actual simulator. Rather, TOSSIM models TinyOS, an embedded operating system. Through the use of the simulator, researchers and developers are able to model different WSN applications on a virtual TinyOS WSN. Although the simulator does not provide real world simulation, it is able to provide an abstraction layer that allows it to simulate the network operation at a low level [12, 13].

Some of the abstraction provided by TOSSIM includes radio propagation within the network. Rather than directly modeling communication and bit errors between two links in a WSN, TOSSIM allows for an external program to model and map a radio model to the bit errors. Additionally, TOSSIM provides power and energy abstraction. The simulator itself does not handle energy consumption and production, rather it elicits
the help of a separate application to apply an energy or power model to the messaging transactions providing a way to calculate the overall energy consumption [12, 13].

While TOSSIM is able to simulate a TinyOS network, it does rely on certain assumptions about the network. As a result, application behavior in the simulation may be different than in the actual network and is not considered authoritative under real world circumstances [12, 13].

2.2 NS

NS or Network Simulator is a discrete event simulator that has been designed to target network research. NS is currently in its third generation, NS-3, development and maintenance is always ongoing although previous versions of NS are still readily available. Released in 2009, NS-3 is an open source simulator. It has been designed around the research community who maintains and develops it. The simulator provides support of TCP, routing and multicast protocols of wired and wireless networks in a controlled environment [14].

The simulator provides control of the various network attributes, giving researchers full control of the simulation. However, with the target audience being a select group, the simulator is a complex application for novice users. Previous generation NS, NS-2 did not provide a full graphical interface and its use was limited to command line prompts. While the NS-3 has been released with a user interface, the experience that it provides is still advanced and can be overwhelming to inexperienced users.
2.3 ATARRAYA

Developed by the Department of Computer Science and Engineering at the University of South Florida, Atarraya is a simulation tool for teaching and researching topology control algorithms for WSN [15]. Topology control is used in a WSN to build a topology with a reduced number of active nodes and links while maintaining network connectivity and area coverage [15].

Although Atarraya users are capable of developing and testing their own protocols, Atarraya is preloaded with various protocols that include topology construction, topology maintenance, sensor-data, and communication protocols [15]. In addition, the simulator is capable of maintaining certain metrics such as energy being consumed and similarly produced. While the simulator is able to randomly deploy nodes, it is a bit harder to statically place nodes in strategic locations.

Similar to TOSSIM it does make certain assumptions regarding items such as the source of energy for certain nodes. Unlike both TOSSIM and NS-2, Atarraya makes use of a visual interface allowing for users to interact with the different components of the simulator. User of the simulator are able to control the number of links as well as the broadcasting and coverage range. Even though Atarraya is capable of providing visual simulation of various protocols, the protocols largely revolve around topology control; the formation and maintenance of the topology. Atarraya is a successful teaching tool for topology control but still lacks depth when attempting to simulate other protocols such as routing.
CHAPTER III

WSN SIMULATOR REQUIREMENTS

Although the individual cost of a single node is relatively inexpensive, building a large WSN for testing purposes can be a costly thing to do. Small networks, while feasible, do not provide adequate results. However, by constructing a simulator for a virtual WSN would save both time and money while giving researchers a better understanding of how different protocols act when given different metrics. Even though there are various simulators that are currently available to simulate a WSN, they tend to focus on a single area of interest such as topology control or for a specific OS. In addition some of the present simulators have a large learning curve that can take weeks to fully comprehend.

In this chapter, a new WSN simulator is proposed that incorporates two different protocols, classic flooding and gossiping. In addition the simulator uses different metrics constraints such as power, mobility, broadcasting range as well as the number of neighbors to construct and simulate a working WSN. The simulator is built to serve both novice and advanced users with the data being easily readable.

3.1 SUPPORTED PROTOCOLS

One of the primary purposes that the simulator will serve is to provide the users with a functional WSN capable of simulating different discovery and routing protocols. When dealing with the initial network deployment; node discovery and network setup phase, the most reasonable discovery protocol is that of the Basic Hello Protocol, BHP.
All of the different discovery protocols that were surveyed essentially have a "hello" phase in which they broadcast a message to their neighboring nodes. While some of the protocols go beyond just a simple "hello," the BHP provides a base for future expansion. Similar to the BHP, the different routing protocols surveyed all share similar routing strategies; WSN nodes route messages from one node to another through multi-hop routing. In a WSN there are two directions that a message will travel within the network; from sink to node and from node to sink. The first type of message are those that are used when information needs to be either broadcasted to the network such as the case if a certain command is to be executed. The second type of message is from the sink to the node, or if a node wishes to send back data or respond to the request of the sink. To simulate these two types of messages, basic flooding and gossiping protocols have been selected to be implemented. These protocols provide the necessary building blocks for future improvement, such as the addition of LEACH and TEEN protocols which not only use multi-hop routing strategies but also a hierarchical model of routing.

3.2 NODE CONSTRAINTS

Wireless Sensor Networks are application specific. Their uses can include area monitoring, intrusion detection to simple surveying. The lifetime of the network itself is also application specific; for certain tasks a WSN may only be needed for a short amount of time while in other cases the network might need to last several months. In order for the network to survive long periods of time there are certain metrics that need to be observed. These metric constraints include but are not limited to power, mobility, transceiver broadcasting range, size of network and the overall network reliability.
3.2.1 Power

The primary cause of the networks lifetime to degrade is the result of bad power management. Power is a commodity resource within the network, a node's power is derived from a battery. With only a limited supply of power, nodes need to be able to manage their power levels very closely. Most nodes are now equipped with a source of renewable energy, typically it is in the form of a solar energy panel. However due to the dynamics of the network, power consumption might be greater than the amount that power is being produced. To properly simulate the lifetime of the network, the user will have control in the simulation to vary both the percentage of power that is consumed every time frame and message sent as well as the percentage of power that is recuperated at every time frame.

3.2.2 Mobility

Node mobility is not an uncommon occurrence within a WSN. Nodes at times may become attached to movable objects such as the case of animal tracking. Other node displacement events may also occur. To account for such movement within the network, users will be given control to indicate the percentage that a node may have of movement within the network. Node mobility will certainly happen, by implementing this in the simulation users will be able to witness how the different protocols and the network itself react to the movement.

3.2.3 Broadcasting Power

A WSN is composed of nodes equipped with transceivers capable of sending and receiving messages. Depending on the size and range of the network, nodes have the
capability of increasing and decreasing their broadcasting power. As a node increases its broadcasting power, the broadcasting range increases allowing them to communicate with distant nodes. However, as the broadcasting power increases, so does the amount of energy that is consumed. Limiting a node’s broadcasting range however alienates their location but increases the life of node, increasing the overall life of the network. A proper simulator needs to grant the user control of the nodes broadcasting range in order to simulate how the power consumption of the transceiver affects the overall network.

3.2.4 Network Size and Reliability

The size of Wireless Sensor Networks can vary from application to application. Typically, the size of the network will depend on the area that the nodes will be deployed in. However the number of nodes that are present in the given region will also determine the network’s reliability.

Having too few nodes results in fewer messages being routed, however it also causes the nodes to increase their broadcasting range thus using unnecessary energy. Furthermore, node movement or failure can cause the network to lose necessary coverage in areas of interest. Having too many nodes can also have a negative effect on the network lifetime. Overcrowding can result in redundant data, excessive messages being routed and packet collisions; all resulting in excessive use of energy.

While the only way to prevent overcrowding of a given region is to deploy the nodes statically, this option is not always a viable one. In addition to static node deployment, nodes can also be deployed randomly. By granting the user the ability to deploy a given set of nodes, successful simulation of various network sizes can be
achieved. To ensure that all parts of a region are covered, the user can additionally deploy static nodes at strategic locations. To ensure proper network reliability the user is also granted control of the number of neighbors that any node can have; this allows for custom settings to improve network reliability.

Using the users settings, different protocols can be tested to evaluate their performance when it comes to message delivery and the amount of messages, similarly the amount of power that is being consumed. In doing so, using these settings helps in deciding the appropriate protocols that can be used under different circumstances. It can also aid researchers in improving current protocols.

3.3 USER INTERFACE

The intended use of this application is to aid both novice and advanced researchers in the visual understanding of a WSN. To accommodate the diverse group, an appropriate user interface is used giving users control of the different options and attributes to simulate the network. Feedback is an important part of the simulation, both visual and concrete data is produced allowing the user to not only continue working on an existing simulation but to also transplant the simulation. This will provide the users with the added understanding of how the network works without having to understand complex jargon which is the case with other simulators currently available.

In order to get the most out of the simulation it is very important that the simulator be available to multiple users. The best way to achieve this is to make sure that the software is compatible across different computers. Windows is the most readily
available platform today, as a result the WSN Simulator uses Microsoft's .NET framework.
CHAPTER IV

WSN SIMULATOR DESIGN

The development of the WSN simulator revolves around providing the user with a fully functional network. The network itself relies on certain tasks that need to be performed and accounted for. They include but are not limited to the initial node deployment, node discovery and the routing protocols as well as power management. This chapter will discuss how each task is handled by the simulator and what possible modifications can be made in order to further improve the accuracy of the results.

4.1 NODE ATTRIBUTES

The main attributes of a node are its Position, a node ID, and its current state. The nodes position indicates where in the deployment region or a Grid a node is located, the Position uses an x-y coordinate system. A node has a unique ID that distinguishes each node apart within a network, while the node state indicates if a node is currently sleeping or awake and able to receive messages. Other attributes within a node are its broadcasting range, its available power supply and a list of neighbors that are used to maintain a node's neighborhood.
<table>
<thead>
<tr>
<th>ATTRIBUTES</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid</td>
<td>Global 2D array representative of the deployment region.</td>
</tr>
<tr>
<td>ID</td>
<td>Unique ID to distinguish a node.</td>
</tr>
<tr>
<td>Position</td>
<td>X-Y Coordinate system of nodes position in the Grid or region of deployment.</td>
</tr>
<tr>
<td>State</td>
<td>An individual nodes current state is either awake and able to participate in routing and discovery protocols or it is asleep.</td>
</tr>
<tr>
<td>Broadcasting Range</td>
<td>Indicates the nodes transceiver broadcasting range.</td>
</tr>
<tr>
<td>Available Power</td>
<td>Indicates the nodes available power supply's percentage.</td>
</tr>
<tr>
<td>Neighbors</td>
<td>A list of nodes that are within range of a broadcasting node used to maintain a neighborhood.</td>
</tr>
</tbody>
</table>

Table 1. List of Important Node Attributes

4.2 NODE DEPLOYMENT

Node deployment is done both statically and randomly. In order to cover a large region, rather than deploying a single node at a time, a random deployment method such as the one in Figure 9 can be used.

```plaintext
Random_Deployment()
for ii ← 0 to Number of Nodes
do
    Node X Position ← Random(0, Width)
    Node Y Position ← Random(0, Height)
while Grid[Node X Position, Node Y Position] != -1
    Grid[Node X Position, Node Y Position] ← ii
    Node ID ← ii
```

Figure 9. Random Node Deployment Algorithm
When the nodes are deployed using the Random_Deployment() function, an x-y position is chosen using the region's width and height attributes. The algorithm will check to see that there are no collisions within the grid, allowing only a single node to reside in a cell. If there are no collisions, the node is added to the grid and an original ID is assigned to the node. Similarly, a node can also be strategically positioned within the grid using the Static_Deploy() method from Figure 10.

```
Static_Deploy(x, y)
    if Grid[x, y] = -1 then
        Add Node
        Node X Position ← x
        Node Y Position ← y
        Node ID ← Node Count
        Node Count ← Node Count + 1
        return true
    else
        return false
```

Figure 10. Static Deployment Algorithm

When a node is to be statically added, the grid cell is checked to see that it is empty. If the cell is empty then a Node can be added with the x-y coordinates provided by the user and an original ID is assigned. The node count is increased before successfully completing the algorithm.

4.3 NODE DISCOVERY

The node discovery algorithm that is used in this simulation is that of the Basic Hello Protocol. This protocol broadcast a "Hello" message, if a node is within the broadcasting range it will acknowledge the message and send back a response. The node that initiated the message will then add the acknowledging node to a list of neighbors.
In this simulation, the user has control of the minimal broadcasting range and the minimal amount of neighbors that a node can have. When a node does an initial broadcast and receives no acknowledgement, the node will increase the broadcasting range till the minimal amount of neighbors are discovered.

```
Node_Discovery ( )
    foreach Node
        x ← Node X Position
        y ← Node Y Position
        for a ← -range do until a > range
            for b ← -range do until b > range
                if INBOUND
                    AND Grid[x-a, y-b] != -1
                    AND distance (x, y, x-a, y-b, range)
                    then Node.Neighbor ← Grid[x-a, y-b]
```

The nodes use a von Neumann Neighborhood to look around the deployment grid. To ensure that a point in the grid is within range, the distance formula in Figure 13 is used.
The distance formula uses the Pythagorean Theorem in Figure 14 to determine the distance of two different nodes. The algorithm will use the x-y coordinates of the broadcasting node and a neighboring node to determine A and B. The distance, C, is then computed. If the distance, C, does not exceed the nodes broadcasting range and the node is currently awake then the node is a valid neighbor. That node is then added to the list of neighbors of the broadcasting node. This process is repeated for all the awake nodes in the simulation.

4.4 FLOODING PROTOCOL

The flooding protocol is traditionally used to broadcast data to the different nodes in the network. The data will disseminate from the sink and will travel across the network to all the different nodes.
Figure 15. Classic Flooding Algorithm

When a node receives a packet of data using the flooding protocol it will retransmit it to all of the node's neighbors. This process is repeated for all of the nodes that receive the message. Since only the intended receiver knows if the message was successfully delivered, the different nodes will continue to flood the network causing excessive packages from being transmitted. To avoid this and to provide the packets with a relatively short time-to-live time a list of visited nodes is added to keep track of nodes that have already received the packets. By adding a list of visited nodes, any infinite loop within the network is prevented. Due to the highly dynamic structure of the resulting graph that the discovery algorithm produces, the flooding protocol is designed to be a Breadth First Search.

4.5 GOSSIPING PROTOCOL

While the classic flooding protocol uses a BFS, the gossip protocol uses a Depth First Search to traverse the different nodes in the network. The protocol is used to disseminate data from the sink to a single node or to send data back from a node to the sink.
The gossip algorithm uses a recursive structure to traverse the different nodes. Once the initial message is deployed from the sink or from a node, the closest neighbor which is typically the first neighbor is chosen to route the message. When a node receives a message, it verifies that it is not the intended receiver, if it is the intended receiver the routing will stop there. If the node is not the intended receiver the node will select a neighboring node and the process will be repeated. To ensure that redundancy is kept to a minimum a list of visited nodes is kept. This ensures that the routing of the message will not cause an infinite loop.

4.6 POWER PRODUCTION AND CONSUMPTION

Power consumption in a WSN has direct implications on the lifetime of the network. Poor power management techniques by the nodes can result in the network exhausting their energy resource rather quickly. Furthermore, aggressive power management can yield poor performance by nodes oversleeping. The main causes of power loss by a node is due to sending and receiving messages.

Figure 16. Gossip Algorithm

```
Gossip(sender, receiver, visited)
    if sender = receiver
        return true
    else
        visited.Add(sender)
        foreach Neighbor in Node
            if NOT visited.Contains(Neighbor)
                if(Gossip(Neighbor, receiver, visited) = true)
                    return true
```
Loss of power due to the amount of messages being sent can be simulated using a formula, such as the one in Equation 4 which has been derived from the nodes broadcasting range and the loss of energy per message being sent.

\[
Energy = Energy - \frac{\text{Broadcasting Range} \times \text{Energy Per Msg}}{100}
\]

Equation 4

Although the formula itself is not representative of the real world energy loss due to message transmission, it does provide a foundation for researchers to derive their own formulas.

Energy loss is greater when sending a message, however just receiving a message causes energy to be used. Equation 5 demonstrates a possible formula to use in the loss of energy due to a message being received. In the equation, only a small fraction of the amount of the energy per message is being lost. Other forms of energy loss come in the form of computation, and node overhead. To account for this type of loss, a new equation is used. Equation 6 demonstrates how much energy is being lost due to overhead per time frame.

\[
Energy = Energy - \frac{\text{Energy Per Msg}}{100}
\]

Equation 5

\[
Energy = Energy - \text{Energy Per Step}
\]

Equation 6

Whenever the energy level of a node drops beyond a certain threshold, the node will enter a sleep state. The node will remain in the sleep state till it regains enough energy to become functional again. Energy in a node is produced primarily through the use of solar panels. The amount of energy that is being produced is limited and is dependent on the node's environment. By using an average node energy production
formula, Equation 7, one can estimate the amount of energy is being produced at every
time frame.

\[ \text{Energy} = \text{Energy} + \text{Energy}_{\text{Produced}} \quad \text{Equation 7} \]

At every time frame, the node will lose power from messages being sent and
received as well as from node overhead. While the amount of energy that the node is
producing might not be enough to permit the node to run constantly, appropriate power
management does allow the node to prolong its life allowing the network to continue to
function.
CHAPTER V

SIMULATION RESULTS AND ANALYSIS

The initial results of the simulator provide promising results. The GUI is able to provide the user with the visualization of the WSN and immediate changes in the different constraints are displayed. The main constraints that the visualization will be looking at are how the nodes are deployed and the appropriate size of the network. The neighbor discovery process and the impact of the number of neighbors have on the network. Furthermore, the protocol being used and the effects that it has on the node's power supply are critical to the overall life of the network.

5.1 NODE DEPLOYMENT

While a traditional network can have up to several thousand nodes present, the simulator is limiting the number of deployable random nodes to 1024. The cap has been placed to allow the simulator to be compatible on multiple machines without the need for high resources. Nodes can be randomly deployed from 2 to 1024 in increasing order or they can be statically placed in strategic locations.

A demonstration using various number of nodes is demonstrated in Figure 17. The number of nodes being used in the figure include 16, 128, 256 and 1024 nodes. For this type of grid, the deployment of too many nodes such as the case with 1024 nodes, 17-D, can result in a cluttered network with too many packet collisions and a high redundancy rate. However too few nodes, 17-A, can result in not enough of the region being monitored. A compromise can be made and a network with 128 nodes, 17-B, in this particular region size can provide adequate coverage by the network.
5.2 NODE DISCOVERY

Working with a network size of 128 nodes, it is easy to visualize how the different nodes can discover their neighbors within a given range. In the algorithm, a node will broadcast a "Hello" packet and expect to receive a response from a neighboring node. However, adjustments can be made to improve the network by adding or taking away redundancy based on the purpose of the WSN.
When the network needs added redundancy, increasing the number of neighbors is done by increasing the broadcasting range. However, when the number of neighbors increases so does the power consumption needed to reach all of the neighbors. This is an added cost that may be acceptable depending on the situation. Similarly by increasing the required number of neighbors can cause a previously disjoint graph to become connected.

**Figure 18. Node Discovery Using Various Number of Neighbors**
For example, Figure 18, demonstrates a region being populated initially with 128 nodes. As the number of minimum neighbors increase, there is a decline of sub graphs and increase in paths from one node to another. Similarly the amount of redundancy of paths in the graph also increase. There is always a balancing that needs to be done when determining what the appropriate number of neighbors needed for each node should be. Based on different graphs that have been generated, a good initial number of minimum neighbors is at least 3; this is subject to change based on the region size as well as the number of nodes in the network.

5.3 FLOODING PROTOCOL

The flooding protocol is used to broadcast messages to all nodes as well as to send a message to a specific node. When a node receives a message it will forward that message to all of its neighbors. This process will continue till the intended node receives the message. However, by that time the message will have spread through the network flooding it with redundant messages. In a highly mobile network, this will ensure that the intended receiver will get the routed packet.
While the flooding protocol does ensure that messages are delivered, it also makes use of excessive network traffic that can quickly deteriorate the nodes power supply. Every node in the network must send the message to all of its neighbors, as seen in Figure 19-B. There is no carrier sensing in a WSN, packet collision and having to retransmit a message will take a toll on the nodes power supply. In this type of protocol, the best case and the worst case scenarios are the same.

5.4 GOSSIPING PROTOCOL

Contrary to the flooding protocol, the gossip protocol is used to transmit a message from a single node to a single source. When a node receives a message, it will forward that message to its closest neighbor. This will continue till the intended receiver is reached. Figure 20 shows an example of the gossiping protocol, where the message keeps getting routed to a random neighbor until the intended receiving nodes is reached. Each node is only aware of their immediate neighbor, so the path to the source may not always be the shortest path.
In the worst case situation, the amount of traffic that is generated by the gossip protocol will resemble the network traffic of the flooding protocol. The best case scenario however will result in a single message being transmitted. Furthermore, the amount of time required to deliver a message will be vary depending on the size of the network.

5.5 POWER CONSUMPTION

The power consumption by the network is dependent on both the protocol being used as well as the node overhead. Depending on the protocol, each node can either send a single message or it can send up to $n$ messages depending on the number of neighbors. Similarly, a node can receive up to $n$ messages depending on the neighbors. Both acts of sending and receiving draw power from the node.
In the above figure, both protocols are used running the same settings: 1\% of the energy is being used up per every message and 2\% of the energy is being restored for every time step. The very first thing that can be viewed is the remaining energy percentage of each node with just one message being sent out by both protocols. There are ways to improve on the amount of energy being used, primarily the flooding protocol would need to be limited to cases of network broadcast while the gossip protocol could be used for communication with the sink or with a single node.
CHAPTER VI
CONCLUSION AND FUTURE WORK

Wireless Sensor Networks are composed of commodity nodes capable of communicating, computing and sensing their surroundings. Although a WSN is very complex and can be difficult to understand, through the use of a simulator it becomes easier to comprehend. Although there are currently WSN simulators available such as TOSSIM, NS and Atarraya, the simulators focus on a narrow topic such as topology control, low level and application simulation. These different simulators provide insight into the different characteristics of a WSN, by building upon their strengths a separate simulator is built to provide the foundation for a fully functional virtual WSN.

In the simulator, users are able to deploy a set of randomly deployed nodes as well as being able to strategically place nodes within a deployment region. The simulator uses the Basic "Hello" Protocol to provide node discovery. Users of the simulator are able to set and modify different attributes of the protocol such as the minimum number of neighbors needed and the broadcasting range of each neighbor. By modifying the different node discovery attributes, different types of networks are constructed and the user is able to control the amount of redundancy that the network provides.

Additionally, the simulator provides two separate routing protocols, Flooding and Gossiping. Users of the simulator are also able to set and modify the routing protocols different attributes such as the directionality of the messages as well as the message sender and the intended receiver. In addition, the user is able to control the amount of
energy being used by sending and receiving messages as well as the amount of energy that is being produced at each time frame in the simulation.

Giving the user control over the different attributes in a WSN, the simulator can not only be used as a teaching tool but it can also aid in the development and modifications of new and existing protocols. The user is able to see the effects that each attribute has on the whole network, from changing the number of neighbors in the discovery process to the amount of energy being consumed.

The protocols provided by simulator, though simple to implement, provide a foundation for future expansion into the simulator. Other discovery and routing protocols surveyed during the research process make use of these protocols to an extent. Hierarchical routing protocols such as LEACH use both multi-hop and single hop routing protocols. By making minor changes to the simulator code, new protocols can be developed and tested.

Additionally, by adding extra functionality such as the sensing range of a node, the user will be able to see the portions of the region that the nodes are able to monitor and should be able to adjust the nodes sensing range to limit power consumption. Though currently a user is able to see statistics from a single node to the entire network, adding a graphical approach such as graphs and charts, as opposed to only tables, would allow novice users to get a better grasp on some of the low level network attributes.
Although the simulator does not focus on a single topic, it does provide a foundation for future work and expansion. While minor changes and improvements will always be needed, the simulator can be considered a successful teaching and researching tool for both novice and advanced users.
CHAPTER VII

REFERENCES


