

Evaluating the Impact of Limited Resource on the Performance of Flooding in Wireless Sensor Networks

Patrick Downey and Rachel Cardell-Oliver
School of Computer Science & Software Engineering
The University of Western Australia
Crawley WA 6009 Australia
pat@graduate.uwa.edu.au
rachel@csse.uwa.edu.au

Abstract—Wireless sensor networks (WSNs) are large collections of resource limited nodes, densely deployed over a landscape. They gather and disseminate local data using multi-hop broadcasting. WSN design and deployment is hampered by currently limited knowledge of the performance characteristics of network nodes and protocols. Their systematic development, thus, requires a flexible simulation environment in which new models of specific node or network behaviours can be integrated easily. This paper introduces a loosely coupled, object oriented simulation environment for this task. The simulator is used to investigate the efficiency of flooding protocols in WSNs. For dense networks with noisy transmission, we show that using low transmission power maximises time and resource efficiency and that the scalability of flooding for large networks is excellent. We demonstrate ways of improving flooding performance given specific deployment constraints.

Keywords— Wireless sensor networks, Flooding protocols, Simulation, Model validation.

I. INTRODUCTION

Sensor networks are a new class of multi-hop broadcasting networks comprising thousands of low cost sensor nodes which sample real-time information across a landscape. Information is communicated from one sensor node to the next using short range radio broadcasting. The new technology of wireless sensor networks enables novel experimental methods for environmental monitoring applications because the relatively low cost of sensor nodes, and their wireless connection, means that properties such as temperature, moisture, or light can be monitored at a finer grain (both in space and time) than has previously been possible.

Traditionally, network designers utilise computer simulations to predict and analyse behaviour. Several widely used network simulators have been extended to model wireless communication by radio broadcast [1], [2] and special purpose simulators have been developed for wireless sensor networks [3], [4], [5], [6]. However, the performance predictions of well known simulators have been shown to differ widely [7], and there are few studies comparing simulation predictions with the observed performance of real networks. In this paper we present a new simulator, designed specifically for wireless sensor networks. Our main goals are transparency and ease of change. Transparency means that the assumptions made in modelling each aspect of network

behaviour from physical radio transmission, to network protocol behaviour, should be visible to the user. Ease of change allows users to substitute their own models in order to obtain greater accuracy, to simplify models where appropriate, or to introduce models based on new experimental results. We show that the object oriented design of the simulator supports these goals. In particular, an event driven interface, the use of reflection, and the ability to dynamically load classes at runtime, make the simulator highly extensible.

Another approach to performance analysis is to implement a sensor network and experiment with different algorithms on the physical system [8], [9], [10]. However, it is time consuming and expensive to build applications and it is difficult to isolate individual properties of an implementation in order to compare different approaches. In particular, the use of the nodes' scarce resources of energy and storage for collecting performance data may create a probe effect, giving distorted data.

The main contributions of this paper are,

- the introduction of an extensible simulation framework and tool for wireless sensor networks;
- an analysis of the effects of limited resources on flooding performance: time and resource efficiency, robustness and energy efficiency.

Section II outlines the Boris simulation framework, and Section III an instantiation of this framework for wireless sensor networks. In Section IV the effects of different parameters on the performance efficiency of flooding is analysed. Section V discusses the validation of the simulator model against empirical data. Related work is discussed in Section VI and conclusions and future work in Section VII.

II. THE BORIS SIMULATION FRAMEWORK

A. Object Model

The object model provided by the Boris simulation framework is a shallow one which allows for maximum flexibility and gives the developer a small hierarchy to learn. The framework itself provides one main class (`Simulator`) that is responsible for the management of high level controls of a simulation: starting, pausing and stopping, and speed. The simulator also provides an event and notification model through the `SimulationEvent` class and

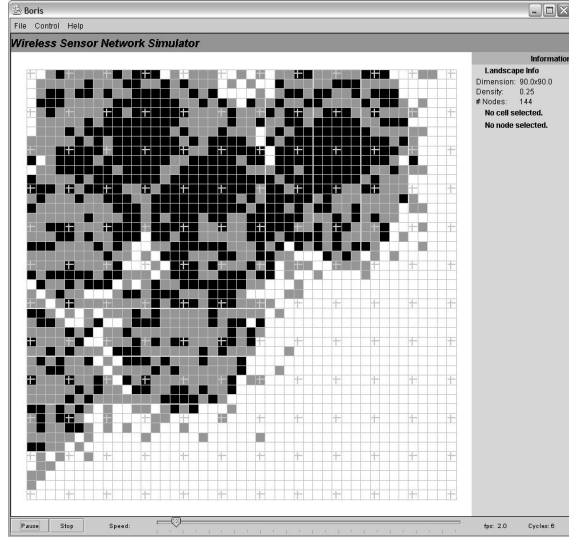


Fig. 1. Boris screenshot showing a flood in progress

`SimulationListener` interface. This model allows for extensions to be developed that can simply plug in to the simulator and react according to any simulation events that occur, such as the completion of a simulation cycle.

One of the benefits of Java, that the framework takes advantage of, is reflection and the ability to dynamically load classes at runtime. This means that classes can be developed and then plugged in to the simulator simply by editing the simulator's configuration file, reducing development time because the developer does not have to register newly developed classes anywhere or recompile any of the simulator's code base. If the new class inherits from the correct parent and compiles, then it can be used.

B. Visualisation

Visualising a simulation is achieved by taking advantage of event notifications provided by the Boris framework. The framework provides a default graphical user interface (GUI) that controls the simulation being visualised, as well as an abstract class that developers should inherit from in order to enable their simulator visualisation class to be embedded in the Boris GUI [11]. The screenshot of Figure 1 shows the encapsulating generic frame consisting of a control panel at the bottom and a menu at the top. The landscape and information bar is provided by a WSN simulator visualisation plug-in.

C. Configuring Simulation Runs

The simulator uses standard Java `System` properties to manage its configuration. The major benefit of using the Java properties API is its native support, which allows configuration to be specified both in a default file and on the command line (through use of the `-D` argument to java), Boris also supports the arbitrary inclusion of a properties

file on the command line. Using Java system properties makes configuration information available for simulator-wide use through the `System` property methods. This allows configuration information to be distributed with ease between sub and peer classes. The use of system properties also allows for simulator extensions to define and use their own configuration keys and values.

D. Simulator Output

The Boris framework and WSN simulator generate output files using the extensible markup language (XML). XML is used because of its descriptive properties which allows for all values to be given an immediately visible context, such as whether an event happened whilst in the send or receive phase. XML is also chosen for the ability to generically parse and/or transform the output. The output can be parsed and transformed by any valid XML parser or extensible stylesheet language (XSL) transformer; all that is needed is the XML schema. The parsing and transformation capabilities of XML allow many different views of the data to be generated relatively easily. The major disadvantage of using XML is the size of an output file. For example the output files generated in the experiments reported in this paper were in excess of four gigabytes.

E. Post Processing Tools

To assist with the analysis of the output data the following tools were developed.

1) *Output Summariser*: To cope with the large amount of data that is generated from multiple simulation runs, a small tool was created to summarise the data from the large output file into a smaller, more manageable XML file. The summary consists of the minimum, maximum, average and standard deviation for each of the following variables: packets sent, packets received, packets processed, and packets dropped.

2) *XSL transformer*: The Boris tools collection also includes a command line XSL transformer which uses the XSL transformer that is built into the Java 2 SDK. This tool was developed to transform the summarised data generated from output from the WSN simulator into files such as a comma separated list of values (for use in a spreadsheet program) or a format ready for plotting graphs.

III. SIMULATING WIRELESS SENSOR NETWORKS IN THE BORIS FRAMEWORK

Our wireless sensor network (WSN) simulator is a finite state machine model that has been implemented on top of the Boris framework. Each cycle in the WSN model consists of three distinct phases:

- 1) *Clear*: Clears the signal state of the landscape,
- 2) *Send*: Processes all nodes waiting to send and
- 3) *Receive*: Processes all nodes waiting to receive a packet.

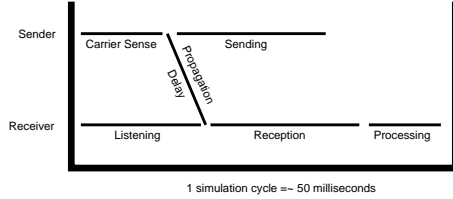


Fig. 2. Events captured by a simulation cycle.

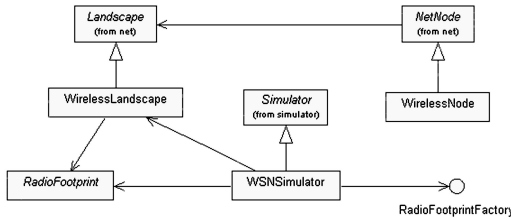


Fig. 3. The WSN object model.

The WSN simulator uses a coarse time scale of one packet transmission interval. Figure 2 shows the events that are incorporated into one cycle of the WSN simulator.

The WSN simulator extends many of the classes provided in the net package of the Boris framework. An overview of the object model and how everything fits together is shown in Figure 3 and further described in the following subsections.

A. Wireless Nodes

Each node in the network is a separate instance of the `WirelessNode` class. This class encapsulates all of a node's operation, such as its routing capabilities and send queue. The node only interacts with the landscape (see Section III-B) through either its `send` or `receive` methods. This partitioning reflects the real life interaction between a node and its landscape.

In our model each node is identified by an integer address and a location, and is comprised of a router, which interfaces with the routing protocol (described in Section III-G), a random number generator, which is used to calculate the back-off, and a number of state variables that keep track of how many packets have been received, sent and dropped.

B. Landscape Model

Network nodes of a WSN reside in a physical landscape. The landscape in the WSN simulator is represented by a configurable two-dimensional grid of *landscape cells* and is responsible for holding all network nodes as well as processing all transmissions and receptions. Each landscape cell is defined by the following data:

- its location,
- a reference to a network node if one is present,
- a reference to a sent packet if one has been received,

- the current radio signal state for the location; one of:
 - 1) `NO_SIGNAL`: if no signal is present.
 - 2) `GOOD_SIGNAL`: if the cell has a good signal.
 - 3) `BAD_SIGNAL`: if the cell has a garbled signal.

The WSN model allows the size and dimension of the landscape to be configured according to how wide and long (in landscape cells) the landscape should be and what the distance between cells should be. This allows flexibility in the resolution of a landscape.

The simulator has been designed to allow for the possibility of extensions, including environmental phenomena such as temperature or humidity (to be recorded by nodes) and other phenomena that may interact with the radio transmission model.

C. Radio Footprint Model

Many algorithms have been proposed for calculating the effect of a radio transmission in a landscape [12], [13], [10]. Some use advanced techniques such as ray-tracing [13] (an extended application from graphics rendering) while others use weighted link states [14], where each node is connected through a virtual wire, and that wire is weighted with a value reflecting the probability of reception. The disadvantages of ray-tracing techniques are their complexity because intimate knowledge of the environment is needed, and their execution speed. Link state techniques on the other hand are fast, but mobile nodes cause complications because of the effort required to keep each node's link weights up to date and interaction with the nodes' landscape is also complicated for the same reasons. In our radio model a radio transmission is represented by a *radio footprint*. When a node transmits a packet, its radio signal propagates in all directions until the signal fades away and disappears. A radio footprint is the entire area covered by such a transmission, and is generally thought to be roughly circular in shape, although current studies [8], [10] show that this is not the case for the low power radio transceivers used in sensor network nodes. The area covered by a radio transmission changes, mostly proportionally, with the amount of power used for the transmission.

The WSN simulator uses an abstract class to represent the footprint of a radio transmission. The class encapsulates a two-dimensional array of the same scale, though not necessarily size, as the parent landscape and is centered around the transmitting node. Each cell in the footprint contains one of three values, `NOISE`, `NO_SIGNAL`, `GOOD_SIGNAL` indicating the type of radio signal present at that cell position.

A table of reception probabilities is used to generate a transmission footprint when a node wishes to send. For a given distance from the sender, the table defines the average percentage of packets which are received at this distance. Table I shows the percentages used in this paper, which are derived from experiments using 110 Rene motes [15], [16]. The table covers reception at distances 1 to 12 cell

TABLE I
PERCENTAGE OF PACKETS RECEIVED VS DISTANCE FOR DIFFERENT
TRANSMISSION POWERS

| Distance (2 ft unit) | Very Low | Low | Medium | High | Very High |
|-------------------------|-------------|-----|--------|------|--------------|
| 1 | 70 | 76 | 81 | 82 | 80 |
| 2 | 63 | 71 | 77 | 80 | 78 |
| 3 | 50 | 67 | 74 | 76 | 74 |
| 4 | 22 | 54 | 66 | 68 | 68 |
| 5 | 5 | 39 | 65 | 65 | 67 |
| 6 | 0 | 20 | 60 | 58 | 65 |
| 7 | 0 | 6 | 48 | 46 | 56 |
| 8 | 0 | 2 | 31 | 32 | 45 |
| 9 | 0 | 0 | 17 | 21 | 36 |
| 10 | 0 | 1 | 9 | 9 | 16 |
| 11 | 0 | 0 | 2 | 9 | 14 |
| 12 | 0 | 0 | 2 | 1 | 2 |

widths (each cell is 2 feet square) at five different power settings. We have averaged a single percentage vs distance, but from the same data we could also derive, and plug into the simulator, a two dimensional model (percentage of packets received at a given x,y offset from the sender) to capture the isometry of footprints [17], or the model could include standard deviations as well as averages [10] to capture the noisiness of radio transmission. We conjecture that the slightly lower percentage of packets received at small distances when using very high power over high power transmission is because high power radio signals tend to distort near to the transmitter. Consider a cell at location (x, y) , which is distance d from a sending node's cell. We choose a random number r . If $(r \leq \text{prob}[d])$ then the footprint cell at (x, y) has value GOOD_SIGNAL. If $r > \text{prob}[d]$ and (x, y) is close to the sender then the footprint cell at (x, y) has value NOISE. If $(r > \text{prob}[d])$ and (x, y) is far from the sender, or outside the footprint range, then the footprint cell at (x, y) has value NO_SIGNAL. In this paper, close and far are defined in terms of a disk centred on the sender: within the disk, each cell that fails to receive a packet contains noise. Outside the disk, each cell that fails to receive a packet contains no signal, and thus is free to receive signals from other transmitting nodes. The disks have radii three to eight for very low to very high power respectively. Further work is needed to validate this model experimentally.

D. Transmission Model

WSN nodes are autonomous and may send at any time. The order in which nodes transmit is important as it affects which nodes can proceed, and which nodes must back off. The set of nodes waiting to send are processed in random order to model the non-determinism of autonomous nodes in physical implementations.

For each node in the landscape that wishes to send, a radio footprint is generated and combined with the current signal state of the landscape according to the rules in Table II.

TABLE II
RADIO FOOTPRINTS IN A LANDSCAPE.

| Initial Landscape | Footprint Cell | Final Landscape |
|-------------------|-------------------|-----------------|
| NO_SIGNAL | NO_SIGNAL | NO_SIGNAL |
| NO_SIGNAL | GOOD_SIGNAL | GOOD_SIGNAL |
| NO_SIGNAL | NOISE | NOISE |
| NOISE | any cell state | NOISE |
| GOOD_SIGNAL | GOOD_SIGNAL NOISE | NOISE |
| GOOD_SIGNAL | NO_SIGNAL | GOOD_SIGNAL |

The effect of these rules is that intersecting footprints will collide where two good signals, or noise, are broadcast to the same cell. In Figure 1 dark cells show transmission noise and collisions, grey cells the good signal, and white cells contain no signal. The position of network nodes is indicated by a cross in a landscape cell. After each simulation cycle all landscape cells are reset to NO_SIGNAL, representing the fading of radio signal from the landscape.

E. Media Access Control

All networks with a shared medium, be it an Ethernet (the wire) or a radio network (the airwaves), have a problem with multiple hosts wanting to transmit simultaneously. The major problem arising from this contention is that the network will become inefficient and congested as the medium becomes full of useless traffic, the result of many transmissions colliding, and subsequently corrupting each other. In networks with a shared medium, hosts need a Media Access Control (MAC) protocol to address this issue of contention. The MAC protocol makes the decision of when a node should transmit a packet, to maximise the probability of the transmission arriving intact at its destination.

A common MAC protocol that is used in wireless networks is the Carrier Sense/Multiple Access (CSMA) MAC protocol. In the non-persistent CSMA protocol [18] used by the WSN simulator, a node that wants to transmit a packet first listens to see if a transmission signal (carrier) is present. If no other transmission is present then the node proceeds with transmission otherwise the node *backs-off* for a random time period, before repeating the procedure. Back-off is the amount of time that the node will wait before attempting to send again. The back-off time in the WSN simulator is bounded by a maximum back-off value that is defined at runtime. The effect of back-off on network behaviour is explored in Section IV-C.

F. Packet Model

The simulator framework provides two abstract classes (Packet and PacketHeader) for the packet model. The WSN simulator extends these classes to create the MediumPacket and DataPacket classes that are used for network communication. The simulator does not use a byte level representation of a transmitted packet, rather these classes are instantiated and shared if necessary.

1) *The Medium Packet*: The medium packet is the ‘envelope’ that each transmission is sent in. Each medium packet has a header and contains a data packet. The header consists of the transmitting node’s address, the destination (next hop) address and a unique packet identifier. A new medium packet is instantiated for every transmission.

2) *The Data Packet*: The data packet consists of a header and a payload, and makes up the body of each data transmission. The header used by the data packet includes the destination and source nodes, and a packet identifier as in TinyOS packets [19]. A packet’s payload is the information that is being transported, typically 30 bytes for TinyOS motes [10]. The classes used for the data packet and header are `DataPacket` and `DataHeader` respectively. For each new data packet that is introduced to the network an instance of `DataPacket` is created

G. Routing Protocol

The aim of the WSN simulator is to model and simulate the behaviour of a wireless sensor network. The major parts of the simulation that define the behaviour of a wireless network is the radio model and the protocols followed by each node. Each node’s protocol is an algorithm which defines what action to take whenever a packet is received or where to direct a packet that needs to be sent. Routing protocols are an important class of protocols which perform, in general, one or more of the following actions after receiving a packet:

- 1) pass the packet to the node for further processing,
- 2) forward the packet to the next hop in the path to its destination or
- 3) drop the packet if it is wayward or too old.

When forwarding a packet, the routing protocol is in charge of deciding the best way to reach a destination.

In an abstract sense, as used by the WSN simulator, the routing protocol is only called when a packet is to be received or when a packet needs to be sent. The Boris framework provides an abstract `RoutingProtocol` class consisting of two abstract methods, `send(DataPacket)` and `receive(MediumPacket)` [11]. The `send` method is called when a data packet needs to be sent and is responsible for wrapping the given data packet in a medium packet and adding it to the node’s send queue. The `receive` method is called whenever a packet is received from the landscape and is responsible for deciding on what action, if any, to take.

The WSN simulator allocates a separate instance of a routing protocol to each node present in the network. This allows for the possibility of different nodes in a network using different routing protocols. The generic `RoutingProtocol` class also allows for developers to implement any other protocols above the data link layer, and link them together as they wish.

The class to use for the routing protocol in a simulation is defined in the configuration file as the value for the

`boris.landscape.nodes.routingProtocol` property. In the experiments reported in this paper a simple *flooding protocol* is used in which each node, with the exception of the flood’s source node, performs the following tasks. The source node starts at step 2, broadcasting a default flood message.

- 1) Wait to receive a packet
- 2) Broadcast that packet as soon as the carrier is free
- 3) No further action

This flooding protocol uses no acknowledgements or collision detection, but relies only on the redundancy inherent in the protocol to ensure that as many nodes as possible receive the flood.

Flooding is chosen for our study because, in a wireless sensor network, flooding protocols are a primitive building block for many other protocols. In addition, the performance of this flooding protocol has been studied in an implemented WSN of 156 nodes [8], and so we are able to compare predicted performance with the observed performance of that network.

IV. PERFORMANCE ANALYSIS OF FLOODING

We have modelled wireless sensor networks in order to evaluate the impact of the limited resources of this type of network on the performance of the network. In this section we present results of a series of simulation experiments to measure the *time efficiency*, *resource efficiency* and *robustness* of a simple flooding protocol in a dense sensor network. For the simple flooding protocol considered in this paper, every node receives and sends a packet exactly one time, and so in our context *energy efficiency* is equal to robustness times transmission power. For the purpose of this experiment we define these parameters as follows:

- **Time Efficiency (cycles)** is the amount of time that it takes for all nodes of a network to receive the flood message (i.e. complete step 1 of the protocol). Where fewer than 100% of nodes are reached by the flood, efficiency is calculated for reached nodes only.
- **Resource Efficiency (percentage)** measures how efficiently the available network resources are used to deliver the flood message. It is defined as the ratio of the time taken for all nodes in the network to receive the flood message, to the time taken for the protocol to complete and all nodes to settle (i.e. complete step 2 of the protocol). Where fewer than 100% of nodes are reached by the flood, efficiency is calculated for reached nodes only.
- **Robustness (percentage)** is the percentage of nodes in a landscape which receive the flood message.

A. Varying Transmission Power

The transmission power used by each node for forwarding the flood packet has a significant effect on the overall lifetime

TABLE III
CONFIGURATION FOR POWER SETTING EXPERIMENTS

| Parameter | Value |
|-----------------|------------------------------------------|
| CONSTANTS | |
| Landscape size | 50 x 50 cells each 2 ft square |
| Node topology | 50 x 50 (1 node per cell) |
| Source Node | (0,0) top left corner |
| Simulation Runs | 100 runs per power setting |
| Maximum Backoff | 5 |
| Footprint Model | Sparse with noise |
| VARIABLES | |
| Power Settings | Very high, high, medium low, very low |

of the network, because each node in a WSN has a very limited energy budget. Using high transmission power, and thus using valuable energy, is only justified if it provides balancing performance improvements in the speed or robustness of flooding. The aim of this experiment is to measure the effects of using different transmission power settings on the overall performance of a WSN. With the larger reach of a high powered transmission it can be reasoned that reception of a flood packet will be faster, but how long will the network be busy reacting to the large amount of activity caused by many nodes transmitting at high power? How does the performance of a flood using low powered transmissions compare with transmissions of a higher power?

The experiment will focus on five transmission power settings: very high, high, medium, low and very low, which correspond to RF transmission powers ranging from 3.25uW (very high) to 0.25 uW (very low) on a Rene mote [20] as used in physical experiments [8].

The parameters used to configure the simulator for the power setting experiments are defined in Table III. These parameters are used to simulate 100 flood runs for each power setting. We report the average number of cycles taken for 25%, 50%, 75%, 95% and 99% of the network nodes to receive the flood and the number of cycles taken for 99% of the nodes to settle. Error bounds (2 standard deviations) are given in brackets for each reported cycle average. Robustness and, thus, energy efficiency were greater than 99% of nodes reached, for all configurations tested, and so are not shown separately.

First we consider time efficiency: how fast the flood spreads at different power levels. At higher transmission power, transmission footprints are larger and so, not surprisingly, the flood spreads more quickly to reach all nodes. The flood initially spreads quickly, but the rate slows dramatically at the end. Reaching the final 5% of nodes takes 15% of the total reception time for very low power to 34% of the total reception time for very high power.

The disadvantage of large transmission footprints can be seen in the final row of Table IV: flooding is much more efficient for very low power than for very high power

transmissions (71% vs 31%). The reason for this is because, for the dense network topology tested, once the flood has reached all nodes, there is a long settling period for high power floods in which nodes that have received, but not been able to transmit, complete their transmissions. The energy spent by nodes during this final settling period is wasted energy because the flood message has already propagated to all reachable nodes.

The total energy expended by network nodes is directly proportional to the transmission energy used by each node since each node receives and transmits exactly once during the flood. Thus, the higher the transmission power, the greater the total energy usage of the network.

The flooding protocol is not robust because some nodes in the network may never receive the flood message. This can occur if every time one or more of a node's neighbours transmit a packet, a collision or corrupted signal occurs at the receiving node's cell. However, the robustness displayed by each of the different power settings tested in this paper is over 99%. For lower power transmissions or less dense node topologies than those reported in this paper, robustness does fall below 99%.

Balancing time and resource efficiency against robustness, we choose to use medium transmission power in our remaining flooding experiments.

B. Varying Landscape Size

WSN applications currently deployed are relatively small, using no more than 100 nodes (e.g. 32 nodes are used at Duck Island [9]), but the vision of sensor network designers is for networks of thousands of nodes, to be used to monitor agricultural, environmental or man-made landscapes. How do basic protocols, such as flooding, scale up in large networks? The purpose of this experiment is to explore the effect of different network sizes on the performance of flooding. For the experiment, three network sizes are chosen: 169 nodes in 13x13 cells, 2500 nodes in 50x50 cells and 10,000 nodes in 100x100 cells. The same node density, of 1 node per 2 foot by 2 foot landscape cell, is used in all three settings. Table V shows the parameters used to configure the simulator.

The time efficiency and resource efficiency for each network size is shown in Table VI. Again, robustness, and thus energy efficiency, is above 99% for all configurations, and so is not shown separately in the table.

Interestingly, the flooding protocol scales very well between network sizes for the dense networks tested here. On average, the flood reaches all nodes in a similar number of cycles for networks of 156 to 100000 nodes. However, the variation in these measurements is much higher for smaller networks. The overall time for nodes to complete the flood is longer for the larger networks and thus resource efficiency is lower. Although the time to reach all nodes by flood is, in principle, proportional to the network diameter (i.e. apx 3, 12, 24 transmissions for the networks tested), the broadcast

TABLE IV
TRANSMISSION POWER VS PERFORMANCE

| | Very Low | Low | Medium | High | Very High |
|------------------|--------------|--------------|--------------|--------------|---------------|
| Received 25% | 12.09 (1.44) | 7.62 (1.26) | 5.66 (1.03) | 5.53 (1.00) | 5.14 (0.70) |
| Received 50% | 16.20 (1.60) | 10.15 (1.28) | 7.68 (1.05) | 7.55 (0.99) | 7.15 (0.71) |
| Received 75% | 19.73 (1.81) | 12.93 (1.42) | 10.93 (1.10) | 10.85 (1.07) | 10.69 (1.19) |
| Received 95% | 25.02 (1.92) | 19.87 (1.67) | 20.85 (2.07) | 21.12 (2.34) | 22.01 (2.27) |
| Received 99% | 29.52 (2.40) | 26.99 (2.52) | 31.64 (3.03) | 32.11 (3.91) | 34.20 (4.04) |
| Settled 99% | 41.37 (1.78) | 51.12 (1.63) | 89.96 (1.74) | 96.33 (2.00) | 111.42 (1.88) |
| Efficiency | | | | | |
| Time E. (cycles) | 29.52 (2.40) | 26.99 (2.52) | 31.64 (3.03) | 32.11 (3.91) | 34.20 (4.04) |
| Resource E. (%) | 71.36% | 52.80% | 35.17% | 33.33% | 30.69% |

TABLE V
CONFIGURATION FOR LANDSCAPE SIZE EXPERIMENTS

| Parameter | Value |
|------------------|---------------------------------------------------------|
| CONSTANTS | |
| Source Node | (0,0) top left corner |
| Simulation Runs | 100 runs per landscape dimension |
| Maximum Back-off | 5 |
| Footprint Model | Sparse with noise |
| Power Setting | Medium |
| VARIABLES | |
| Landscape sizes | 13 x 13, 50 x 50, 100 x 100 cells each cell 2 ft square |
| Node topologies | Square grids of 169, 2500, and 10000 nodes |

TABLE VI
NETWORK SIZE VS PERFORMANCE

| Network Size | 156 nodes | 2500 nodes | 10000 nodes |
|------------------|---------------|--------------|--------------|
| Received 25% | 1.91 (0.57) | 5.59 (0.98) | 10.07 (0.88) |
| Received 50% | 2.55 (1.11) | 7.66 (0.99) | 13.56 (1.03) |
| Received 75% | 5.04 (2.21) | 10.84 (1.05) | 17.23 (1.12) |
| Received 95% | 17.25 (9.50) | 20.73 (1.96) | 26.95 (1.36) |
| Received 99% | 35.86 (15.35) | 31.36 (3.27) | 37.17 (1.70) |
| Settled 99% | 68.45 (4.35) | 89.92 (1.62) | 97.13 (1.34) |
| Efficiency | | | |
| Time E. (cycles) | 35.86 (15.35) | 31.36 (3.27) | 37.17 (1.70) |
| Resource E. (%) | 52.39% | 34.88% | 38.27% |

storm problem [21] appears to be more pronounced in smaller networks, and so, proportionally, flooding performs worst in small scale networks.

C. Varying Maximum Back-off

The amount of time that a node backs off before attempting to send can have a significant effect on relieving network congestion and improving network throughput as a result. The aim of this experiment is to look at the performance improvements possible in WSNs by adjusting MAC layer back-off.

As can be seen in Table VIII, different values of max-

TABLE VII
CONFIGURATION FOR MAXIMUM BACK-OFF EXPERIMENTS

| Parameter | Value |
|-------------------|--------------------------------|
| CONSTANTS | |
| Landscape size | 50 x 50 cells each 2 ft square |
| Node topology | 2500 nodes, 1 per cell |
| Source Node | (0,0) top left corner |
| Simulation Runs | 100 runs per back-off |
| Footprint Model | Sparse with noise |
| Power Setting | Medium |
| VARIABLES | |
| Maximum Back-offs | 0, 2, 5 |

TABLE VIII
MAC LAYER MAX-BACKOFF VS PERFORMANCE

| | maxb=0 | maxb=2 | maxb=5 |
|------------------|--------------|--------------|--------------|
| Received 25% | 5.65 (0.95) | 5.63 (0.97) | 5.57 (0.99) |
| Received 50% | 7.83 (0.98) | 7.83 (0.90) | 7.66 (0.99) |
| Received 75% | 12.53 (1.40) | 11.85 (1.37) | 10.81 (1.08) |
| Received 95% | 27.55 (2.76) | 24.91 (2.74) | 20.90 (1.82) |
| Received 99% | 41.91 (4.05) | 38.21 (3.97) | 31.41 (3.02) |
| Settled 99% | 78.91 (1.33) | 82.21 (1.45) | 89.87 (1.78) |
| Efficiency | | | |
| Time E. (cycles) | 41.91 (4.05) | 38.21 (3.97) | 31.41 (3.02) |
| Resource E. (%) | 53.11% | 46.48% | 34.95% |

imum back-off have little effect until 75% of nodes have received the flood. However, there are significant time and resource efficiency differences. Resource efficiency decreases as maximum back-off increases because the number of cycles used to cover the final 5% is stretched when larger back-offs are used. On the other hand, time efficiency improves for larger back-offs because contention is reduced while the flood message is propagating. Thus there are trade-offs to be made in selecting the best performance based on these two performance indicators. Robustness is greater than 99% for all levels; reducing contention by increasing back-off has a minimal effect on robustness for this network configuration.

D. Quantifying the Effect of Limited Resource

Our goal is to analyse the effect of the limited resources of a wireless sensor network on the performance of that network. In this section, we consider the effect of limited radio transmission resource at each node. We compare the performance of the flooding algorithm using

- *sparse footprints*, as occurs in actual networks, with
- *idealised, dense footprints*, as used in many models of wireless networks.

For comparison we chose dense footprints which reach exactly the same number of cells as their (average) sparse counterparts. For example, the medium sparse footprint reaches, on average 113 cells, within an overall transmission radius of 12 cells. Its corresponding dense footprint is a disk of radius 6 cells with no noise. Table IX summarises the differences in network performance for these two footprint models. At very low and low powers, noisy footprints spread the flood faster and settle faster, than flooding with an idealised, dense footprint. However, at medium, high and very high transmission power, noisy footprints are slower to spread the flood, and slower to settle. Although the resource efficiency with noisy footprints is usually higher than with ideal footprints, this only occurs because time efficiency is much lower, and so the overall performance of observed footprints is worse than for idealised. That is, at low transmission powers, the resource limitation of noisy footprints in wireless sensor networks actually improves the performance of flooding, whilst at high transmission power, this resource limitation leads to worse performance.

V. VALIDATION OF THE SIMULATOR MODEL AGAINST EMPIRICAL DATA

In this section we compare our simulation results on the performance of flooding in WSNs with those with an empirical study of flooding in a 156 node sensor network [8]. These two studies have similar findings:

- As transmission power increases from very low to very high, *time efficiency* increases too. Equivalently, *reception latency* increases [8].
- As transmission power increases from very low to very high, *resource efficiency* falls. Equivalently, as transmission power increases the percentage of *useless retransmissions* rises [8]. Efficiency falls from around 70% to 30% in both studies.
- Robustness is over 99%, but not 100%, for a dense network topology of 1 node per 2ft by 2ft cell for all transmission powers. The percentage of nodes which received the flood is not reported in the empirical study [8] but was most likely 100% for the 4 to 7 runs observed at each power setting.

However, there are also some differences between the results, for example in the 95% confidence intervals for timing (in milliseconds in Ganesan *et al.* and cycles for the

simulator). For medium transmission power, considering 95% of the network nodes, the error observed in experiments [8] is 18% for reception time and 4% for settling from 7 runs with a 156 node network. For the same results in simulation the error is 55% for reception time and 5% for settling from 100 runs with a 169 node network. The small network is particularly unstable, with significant performance differences depending on the order in which nodes transmit, which may account for the larger errors measured in 100 simulation runs. The performance of larger networks is more stable.

Our simulations use an average footprint for each power setting, derived from Woo's experimental data for 110 of the nodes used in the flooding experiments [15], [8]. Interestingly, for the same power setting, the footprint size of individual nodes as observed in experiments [15] can vary dramatically. For example, averaging over the 110 nodes whose footprints were measured at medium power, the average number of footprint cells containing good signal is 113. The number of good cells in a full footprint was inferred from the experimental data when the full footprint was not directly measurable because it would fall outside the experiment's landscape. At medium power, the worst performing node's footprint contains an average of 47 good cells. The best performing node's footprint contains an average of 297 good cells. Obviously, using the average footprint for all nodes can be misleading. One solution is to build standard deviations into the footprint calculation algorithm [10]. However, since the difference between footprints seems to be associated with particular node hardware and position, rather than simply noisiness in each node's performance, that solution does not adequately capture the diversity of node performance. Further work is needed to resolve this issue. Node diversity may turn out to be much less significant for the more reliable radio hardware in current generation Mica 2 nodes [20], [17].

VI. RELATED WORK

Our simulation experiments were motivated by a recent empirical study of the performance of flooding protocols in wireless sensor networks [8], [15]. This study used a sensor network consisting of 156 Berkeley nodes [20] placed in a regular grid pattern in a 26 foot by 26 foot landscape. Each node in the network, in turn, transmitted 20 packets at different transmit power settings. The number of these packets received by each surrounding node is recorded [15]. In a second experiment, a flood is initiated from a node in the middle of the base of this grid, and each node recorded its transmission and reception of messages during the flood. Using raw data from the footprint experiment [15], we can estimate the probability of reception given distance and direction from the source. We used these probabilities in our simulation experiments to define transmission footprints. We also used the same network topology, transmission powers, and flooding protocol as the empirical study.

TABLE IX
PERFORMANCE COMPARISON OF IDEAL VS OBSERVED FOOTPRINTS

| | | Very low | Low | Medium | High | Very high |
|---------------------------------|--------------------------|----------|--------|--------|--------|-----------|
| Ideal Dense Footprint | Time efficiency (cycles) | 39.93 | 27.34 | 21.27 | 21.41 | 20.17 |
| | Cycles to finish | 50.64 | 51.79 | 78.09 | 82.88 | 95.68 |
| | Resource efficiency | 78.85% | 52.79% | 27.24% | 25.83% | 21.08% |
| Observed Sparse Footprint | Time efficiency (cycles) | 29.52 | 26.99 | 31.64 | 32.11 | 34.20 |
| | Cycles to finish | 41.37 | 51.12 | 89.96 | 96.33 | 111.42 |
| | Resource efficiency | 71.36% | 52.80% | 35.17% | 33.33% | 30.69% |

As well as empirical results, there are several analytical studies of flooding algorithms relevant to our investigation. A study of the *broadcast storm* problem for ad hoc networks, of which sensor networks are an example, is presented by Ni *et al* together with several variations on the basic flooding protocol [21]. Geometric analysis is used to investigate broadcast redundancy, contention and collisions, and simulation to analyse the performance of flooding in a network of mobile nodes with different node densities. A study by Sasson et al [22] investigates probabilistic flooding, looking for phase transitions, again using an ideal circular broadcast footprint. A simulation model for wireless sensor networks which incorporates a noisy footprint model is presented in [6] and used to analyse the performance of protocols such as flooding in wireless sensor networks. The simulator searches for configurations which optimise certain properties, such as maximising reliability whilst minimising consumed power. Analytical methods based on differential equations have been used to create epidemiological models for the spread of disease and such models can also be used to analyse flooding algorithms [23]. As in the broadcast storm study, a circular transmission footprint is assumed and different densities of network nodes are considered. In addition, nodes are mobile, and follow a random movement model. The epidemic model describes the *infection* of all nodes, which is comparable to time efficiency in this paper. However, it does not allow the protocol followed by the nodes to be changed, different transmission footprints to be used, nor does it allow the measurement of settling time.

Woo, Tong and Culler [10] investigate the performance of routing protocols, including flooding, using a Matlab simulator which takes into account footprint noise in a similar way to our model, and also using a 50 node experimental network. However, they compare the performance of different routing protocols in the same WSN setting, whereas we examine the effects on performance of underlying characteristics such as landscape size, transmission power and MAC layer constants.

VII. CONCLUSIONS

We have presented an object oriented simulation framework, Boris, and described its instantiation as a wireless sensor network simulator.

Our simulation experiments support the following observations on the impact of limited resources on the performance of flooding in a wireless sensor network:

- Low power transmissions have the best time and resource efficiency for the dense network topology we simulated. They are also the most energy efficient for the network as a whole and for each individual node.
- The flooding algorithm has excellent scaling properties for large networks. Most activity occurs in parallel, and the only bottle-necks are local effects where network nodes in the same transmission neighbourhood must wait for others to transmit before they can obtain the broadcast medium. The performance of large networks is more stable than that of small networks. For example, the number of cycles for 95% reception time has errors of 5% to 9% for large networks as opposed to 55% for a small network.
- The performance of flooding algorithms is readily tunable, even given a fixed node topology, because variables such as transmission power and MAC layer backoff are all accessible to the developer. We have shown that the adjustment of maximum backoff, for example, can be used to improve protocol performance.
- The flooding protocol is over 99% robust in dense networks for all configurations tested. This is despite the noisiness of low power radio footprints, and the high probability of collisions between competing transmissions from overlapping footprints.
- The resource limitation of noisy footprints vs idealised dense footprints in wireless sensor networks has a significant impact on performance. At very low transmission power, noisy footprints actually improve the performance of flooding, whilst at all other transmission powers, this resource limitation leads to worse performance than with idealised footprints. This result demonstrates that simulation experiments for investigating performance should take into account the observed transmission footprints of motes and not rely on idealised transmission models.

Our study has highlighted several areas for further work. There is clearly a need for more empirical data from sensor network applications in order to develop accurate models of

different aspects of network performance, and so to establish a systematic basis for protocol design and deployment.

We also plan to add classes to the simulator to model the energy available to each sensor node, to model environmental factors such as the behaviour of the environment being sensed, and to investigate higher level protocols (than flooding) such as energy aware routing protocols.

VIII. REFERENCES

- [1] Information Sciences Institute, *The Network Simulator - ns-2*. University of Southern California, 2003. [Online] Available at <http://www.isi.edu/nsnam/ns/> as of 28 October 2003.
- [2] X. Zeng, R. Bagrodia, and M. Gerla, "Glomosim: A library for parallel simulation of large-scale wireless networks," in *Workshop on Parallel and Distributed Simulation*, pp. 154–161, 1998.
- [3] P. Levis and N. Lee, "Nido system description." <http://webs.cs.berkeley.edu/tos/tinyos-1.x/doc/nido.pdf>, Accessed August 2003.
- [4] L. F. Perrone and D. M. Nicol, "A scalable simulator for tinyos applications," in *Proceedings of the 2002 Winter Simulation Conference* (E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.), 2002.
- [5] J. Elson, S. Bien, N. Busek, V. Bychkovskiy, A. Cerpa, D. Ganesan, L. Girod, B. Greenstein, T. Schoellhammer, T. Stathopoulos, and D. Estrin, "Emstar: An environment for developing wireless embedded systems software," 2003. CENS Technical Report 0009, March 24, 2003.
- [6] G. Simon, P. Volgyesi, M. Maroti, and A. Ledeczi, "Simulation-based optimization of communication protocols for large-scale wireless sensor networks," 2003. <http://www.isis.vanderbilt.edu/projects/nest/prowler/>.
- [7] D. Cavin, Y. Sasson, and A. Schiper, "On the accuracy of manet simulators," in *Principles of Mobile Computing 2002, Toulouse, France*, 2002.
- [8] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker, "An empirical study of epidemic algorithms in large scale multihop wireless networks," 2003. Submitted for publication.
- [9] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, (Atlanta, GA), Sept. 2002.
- [10] A. Woo, T. Wong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *ACM Sensys 03*, 2003.
- [11] P. Downey, "The behaviour of a flooding protocol in a wireless sensor network," 2003. Honours Thesis, School of Computer Science & Software Engineering, The University of Western Australia.
- [12] K. Fall and K. Varadhan, eds., *The ns Manual*. University of Southern California, 1999. [Online] Available at <http://www.isi.edu/nsnam/ns/doc/> as of 28 October 2003.
- [13] J. W. McKown and R. L. Hamilton, Jr., "Ray tracing as a design tool for radio networks," *IEEE Network Magazine*, vol. 5, pp. 27–30, November 1991.
- [14] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: Accurate and scalable simulation of entire tinyos applications," in *To appear in Proceedings of the First ACM Conference on Embedded Networked Sensor Systems*, University of California, Berkeley, 2003. [Online] Available at <http://www.cs.berkeley.edu/~pal/pubs/tossim.pdf>.
- [15] A. Woo *et al.*, "Connectivity experiment." [Online] Available at <http://www.cs.berkeley.edu/~awoo/connectivity/> July 2003.
- [16] R. Cardell-Oliver, "Analysis of experimental transmission footprint data." [Online] Available at <http://www.csse.uwa.edu.au/~rachel/DSP04/> December 2003.
- [17] A. Cerpa *et al.*, "Scale: A tool for simple connectivity assessment in lossy environments." Draft paper from Alberto Cerpa, 3rd September 2003.
- [18] A. S. Tanenbaum, *Computer Networks*. Prentice-Hall International, third ed., 1996.
- [19] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, "System architecture directions for networked sensors," in *Architectural Support for Programming Languages and Operating Systems*, pp. 93–104, 2000.
- [20] "Mica motes." www.xbow.com/Products/Wireless_Sensor_Networks.htm, Accessed May 2003.
- [21] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, pp. 151–162, ACM Press, 1999.
- [22] Y. Sasson, D. Cavin, and A. Schiper, "Probabilistic broadcast for flooding in wireless mobile ad hoc networks," in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC 2003)*, 2003.
- [23] A. Khelil, C. Becker, J. Tian, and K. Rothermel, "An epidemic model for information diffusion in manets," in *Proc. Fifth ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, September 2002, Atlanta, Georgia, USA*, 2002.