

# Detailed Models for Sensor Network Simulations and their Impact on Network Performance \*

Maneesh Varshney  
Computer Science Department  
University California, Los Angeles  
Los Angeles, CA 90095-1596  
{maneesh, rajive}@cs.ucla.edu

Rajive Bagrodia  
Computer Science Department  
University California, Los Angeles  
Los Angeles, CA 90095-1596  
rajive@cs.ucla.edu

## ABSTRACT

Recent trends in sensor network simulation can be divided between less flexible but accurate emulation based approach and more generic but less detailed network simulator models. We offer an approach with the flexibility of network simulators and provides the accuracy comparable to emulation based approaches. We describe the design and architecture of sensor network simulator which provides a rich suite of following models: sensing stack to model wave and diffusion based sensor channels, an accurate battery model, processor power consumption model, energy consumption model and sensor network based traffic model. We also present our study on the effects of detailed modeling on the performance of higher layer protocols. We describe the impact of using accurate models for battery, processor power consumption and traffic models on the network layer statistics as network lifetime and availability, throughput and routing overhead. Our results show there is high sensitivity of model accuracy on network and application level statistics. In extreme cases we have noticed an inversion of results with our accurate models as compared with previous generation of abstract models.

## Categories and Subject Descriptors

I.6.5 [Computing Methodologies]: Simulation and Modeling—*Model Development*; I.6.4 [Computing Methodologies]: Simulation and Modeling—*Model Validation and Analysis*

## General Terms

Performance, Design, Experimentation, Verification

## Keywords

Sensor network, simulation

\*This work is funded by NSF under NRT grant ANI-0335302

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MSWiM'04, October 4-6, 2004, Venezia, Italy.  
Copyright 2004 ACM 1-58113-953-5/04/0010 ...\$5.00.

## 1. INTRODUCTION

The advances in MEMS based sensor devices and miniaturization of processor and radio as sensor node package has led to emergence of sensor networks. Interesting scenarios and applications can be conceived that deploy these nodes in a field where they work collaboratively to sense the environment, perform in-network computation or storage and communicate with monitoring station if and when interesting events occur. As they are envisioned to be deployed in large-scale it is necessary that the development phase be preceded by simulation to develop prototype, validate and compare performance.

The network simulation for sensors is a challenging problem as it has to faithfully model the constraints of hardware and energy, which is typical with sensor nodes and also have to model various aspects exclusive to sensor networks. Current efforts in sensor network simulation can be divided in two camps. First category is the emulation-based approach, which provides accuracy in certain aspects of modeling at the cost of flexibility and completeness of simulation. The primary purpose is to help in early prototyping and debugging of applications. The other approach is close to network simulations, that focuses on scalable and modular design but do not model the various components accurately. We present our work as a bridge between the two ends: we provide a network simulation oriented approach that includes a rich set of detailed models.

In this paper we present the sensor network simulator architecture that provides support for sensing capabilities in network nodes, sensing channels that include wave and diffusion based propagation, an accurate and efficient battery model, processor model to account for the delay and energy consumed by execution of code, power consumption model to faithfully represent an actual sensor hardware and sensor network oriented traffic model.

We also show the impact of these detailed models on the performance of network and application layer protocols. This is similar to the work done in [11] which evaluated the effects of utilizing details in modeling in context of ns-2 simulator, and in [25] which evaluated the effect on the performance of higher layer protocol by detailed physical models, in context of GloMoSim simulator. We have studied sensor network models in the context of network simulation and this is the only work to our knowledge that compares the effect of detailed models on the performance of sensor networks. The following three case studies illustrate the importance of using accurate and representative models.

1. We use an accurate battery model to study the transient behavior of a network when its power supply, as represented by the aggregate battery lifetime of all its nodes is being depleted. In particular, we show that route fluctuations and routing overheads are very sensitive to the accuracy of the battery model; consequently the bit rate throughput and application layer metrics are impacted in a non-linear manner by battery model used in the simulation.
2. Accounting for the processor power consumption to measure the power consumption of nodes in the network have shown important trends in application level performance that are not visible when using only radio power model.
3. Using a realistic sensor network traffic model had a dramatic impact on the relative performance of different routing protocols. With equal offered load in the network and using random source-destination pair CBR traffic we found performance of AODV and DSR protocol to be very similar. However, with a traffic mode that was more representative of sensor network we discovered that DSR outperforms AODV by as much as 500%.

To summarize, the contribution of our paper is two-fold: we present a rich suite of detailed models for sensor network simulation and we show the impact of detailed models on the predictions of network performance when using these models.

The rest of the paper is organized as follows: section 2 discusses the related work in modeling of sensor networks, section 3 describes the simulation architecture and models, while the quantitative effects on performance of network and applications of these models are shown in section 4. We conclude in section 5.

## 2. RELATED WORK

**Sensor Network Simulation:** Existing work on sensor simulation can be categorized under four sections: using pure network simulators, extensions to network simulators, building simulators from-the-scratch and emulation of sensor hardware. The commonly used network simulators are ns-2 [17], GloMoSim [29] and its successor Qualnet [5] etc. These simulators have been enriched with sensor network specific MAC [27] and network [12] protocols, but they do not model any other aspect of sensors.

SensorSim [19] extends the ns-2 network simulator with models of sensor channels, accurate battery and power consumption. Each node has a sensor stack that acts as a sink to the signals in the sensor channels or can generate such signal. This is closest to our work in terms of philosophy of sensor network simulator architecture. However, we have extended the features by including diffusion sensor channel model, code execution analysis and a more accurate battery model.

The whole class of start-from-scratch simulators focuses on the ability to simulate very large number of nodes and is less concerned with providing detailed models. Sensor, Environment and Network Simulator, SENS [24] is a platform independent, modular and scalable architecture for modeling the network communication, applications and environment. The various entities are modeled as components in

a graph and these components can communicate. Direct portability of sensor code is possible by linking with the application API library. SENSE [6] is a simulator that focuses on providing extensibility, scalability and reusability. They have used a different architecture for simulation: the component port model, where different components can be interfaced easily. The work on models is still in progress, the existing suite is simplistic even compared with network simulators. SensorSimII [26] also aims at providing modular architecture. The wireless protocol stack is borrowed from network simulators and the applications can respond to triggers from sensor component. Simulation of Wireless Adhoc Networks, SWAN [15], is an effort toward simulation of very large networks. They have used DaSSF, Dartmouth Scalable Simulation Framework [16] as base architecture in which they have added the wireless protocol stack and sensing model.

TOSSIM [13] is an emulator of TinyOS for the mica motes [8]. The emphasis of this tool is to be able to develop and debug the applications targeted for mica motes. The network support is very simple. The simulator assumes that all nodes are running the same code. TOSSF [20] extends the simulation functionality by adding the SWAN scalable framework into it. Atemu [1] is an emulator for AVR based processor. It completely emulates the mica2 board and provides simple channel model for simulation.

**Battery Models:** Perhaps the most accurate model for battery behavior is due to Newman [10]. They consider the microscopic properties of battery elements and solve a set of partial differential equations numerically. A more abstract model include [18], which use markov model to predict the lifetime of battery. In between these extremes is diffusion based model from Vrudula and Rakhmatov [22] in which they have abstracted two parameters to characterize a battery: capacity and non-linearity. We have used this model in our implementation of battery model, but have optimized it for the special case when the loads are of small magnitude and duration.

**Processor Power Consumption:** JouleTrack [23] program takes as input a program and target processor type. It cross compiles the code into that architecture and emulates the code to output the execution time and power consumed in executing the code. They have created a database for power consumption by each instruction type for StrongARM processor [3]. This is a standalone program, and it is not easy to see how it can be integrated with network simulators. A more detailed analysis of power consumption is by SimplePower [28] tool that use SimpleScaler [7] simulator of target processor that considers memory interaction, cache miss etc. This is too slow and probably too detailed to be used in network simulations.

## 3. SIMULATOR ARCHITECTURE

In this section we describe the various models that have been added in Qualnet network simulator. Fig 1 outlines the sensor node architecture.

We can divide the component diagram in five broad categories: (a) *Sensing Channels and Stack* (Sec 3.1): An exclusive feature of sensor networks is their ability to sense the environment. Here we describe the models we have added for wave-propagation and diffusion based sensor channels; (b) *Battery Model* (Sec 3.2): we use an analytical battery model [22] and describe our optimization that improved the per-

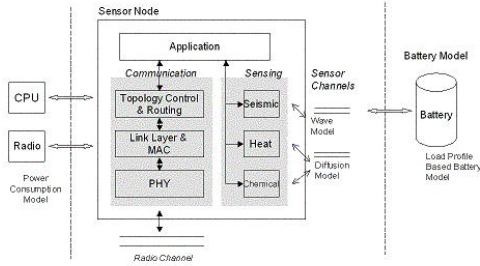


Figure 1: Sensor Node Architecture

formance by three orders of magnitude; (c) *Processor Model* (Sec 3.3): we provide a non-emulation based approach to account the power consumed and delay introduced when executing the protocol and application code by the processor. We describe our methodology for StrongARM SA1100 processor, which is used in WINS sensor node [9]; (d) *Power Consumption Model* (Sec 3.4): describes the mechanism to model power consumption for any number of devices based on actual hardware specifications, and the agility to easily switch from one architecture model to another; and (e) *Communication Stack and Traffic Model* (Sec 3.5): we describe a traffic model that represents sensor network traffic more closely.

### 3.1 Sensing Channel Model

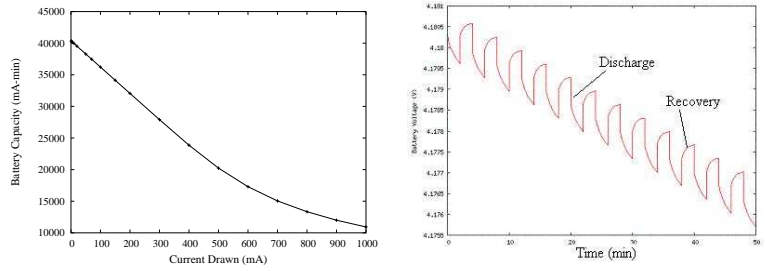
A typical sensor node is equipped with one or more of the following sensors: seismic (acceleration), acoustic, heat, magnetometer, pressure, chemical etc. Each of these types senses some phenomenon that propagates either through *wave mechanics* or *diffusion*. The former follows the inverse distance power law and suffers from path-loss, fading, multipath etc (e.g. sound waves, seismic waves). The latter can be defined as the property of movement of specie across a gradient from region of low concentration to high (e.g. heat, chemicals in air). We have implemented both of these propagation models (channels), which we describe next, as building blocks upon which any specific sensor channel can be defined.

The wave propagation channel is implemented similar to the radio propagation models of Qualnet simulator. For diffusion channel we have used Fick’s law for temperature gradient and Fourier’s law for concentration gradient.

Whereas the two channel models described above are global data structures, a *sensor phy* is defined per node. It is that part of sensor node which interacts with channel model to read the value, provides API to the application layer and can act as actuator to influence the environment.

### 3.2 Battery Model

An ideal battery is usually viewed as reservoir of charge from which an amount equal to the load can be subtracted, until the capacity falls to zero. However, this is not an accurate model as a real life battery shows *non-linear* discharge behavior and *recovery* effects [14]. Non-linearity implies that the battery drains at increasingly faster rate when higher loads are applied. The capacity of the battery is dependent on rate of discharge, as shown in fig 2(a), which should have been a horizontal line in an ideal case. A battery can also recover some of its capacity lost earlier, if it is allowed a rest



(a) Non-linearity.

(b) Recovery.

Figure 2: Real battery behavior.

period or discharged at lower rate. This is shown in fig 2(b), where battery goes through alternating states of discharge at 600mA for 2 min and rest period of 2 min. Data for both is collected using Dualfoil program [2] for lithium-ion battery using the default parameters.

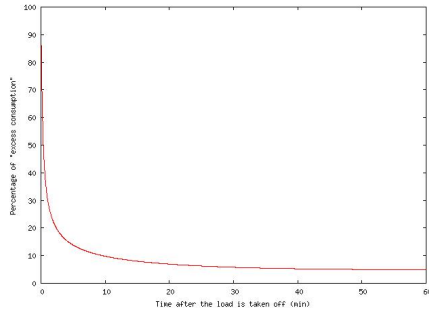
In order to model the battery behavior accurately we have used the model due to Vrudhula and Rakhmatov [22], in which they have explained the non-linearity and recovery by process of diffusion of electrolyte. Let  $I(t)$  is the current drawn from the battery at time  $t$ , then the battery capacity used after time  $T$  is given by  $C(T) = \int_0^T I(t)dt + 2 \sum_{m=0}^{\infty} \int_0^T e^{-\beta^2 m^2 (T-t)} dt$ . The lifetime  $L$  of the battery is obtained by solving the equation  $\alpha = C(L)$ . Note that the first term in the equation is the ideal power source consumption while the second term represents the non-linearity and history based recovery.

However, this equation is inefficient to simulate packet level events. The model was primarily designed for user applications running on PDAs where load profile lengths are 10-100, have large current values and running time of applications in minutes (as reported in their experimental validation data). For network level simulations, the duration would be milliseconds (time to transmit a packet, for example), small current drawn (low power radios in sensor nodes) and the profile length is in order of 100s of thousands. Our first attempt to use the equation described above directly failed, as it was very slow. We describe next our optimizations, in which we observed the behavior of the equation under the case when the loads applied are low and for very small durations and found that we can achieve a performance gain of three orders of magnitude with the optimizations.

**Model Optimization:** If current  $I$  is applied for duration  $T$ , then the total battery consumption can be written as  $C = I*T + L$ , where  $I*T$  is the ideal discharge and  $L$  is the excess discharge. The value of  $L$  is maximum at time 0 (defined as time when the load is taken off from battery) and decreases in magnitude with time, and finally goes to zero after sufficiently long time.

The value of  $L$  at any time  $t$  can be obtained if we have following two pieces of information: the maximum excess consumption,  $L_{max}$  and the rate at which it decreases with time. We have observed that for a given value of current the value of  $L_{max}$  depends linearly on duration  $T$ , for small values of  $T$ . If we know beforehand what are the possible values of current drawn from battery, we can store this function as pre-computed result.

The rate of decrease of  $L$  is shown in fig 3. We have plotted the percentage excess consumption, defined as  $100*L/L_{max}\%$ .



**Figure 3: Recovery of "excess" consumption with time**

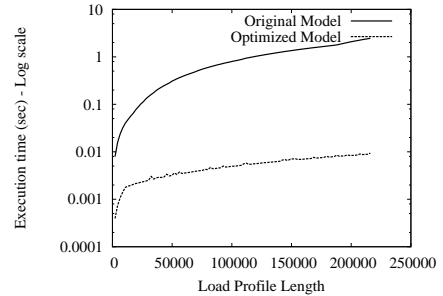
We observed that the curve is same for all values of  $T$ , if it is small. We fitted a 8th-degree polynomial function with this curve and store it as pre-computed result in our simulation. During runtime we use these two functions to evaluate the capacity consumed. Fig 4 compares the performance of our optimized model with the original model, where we see improvement of over 500 times while within 0.1% error.

### 3.3 Processor Model

The sensor nodes are equipped with very low-end processors or micro-controllers [8], [9] and therefore, would take non-negligible time to execute the protocol and application code. Moreover the power consumed could be a significant proportional of total power consumed. It is, for these reasons, necessary to also model the behavior of code executed by the processor. However, we do not like to take the pure emulation based approach (e.g. [13], [20], [1]), as emulation becomes bottleneck for scalable simulations and it restricts the simulator for one sensor architecture only.

We propose an approach that differs in two ways. We do not use any particular hardware specific code, but work with the code in the simulator. That is, we assume that the protocol and application code written in simulator is close approximation to the actual code. In this we tradeoff between the accuracy obtained by utilizing specific hardware optimized code with the flexibility to use the same code for multiple platforms. The other distinction is that we use emulation of target platform (the sensor processor) only for precomputing results. At runtime, the simulation is carried out in native code. In the precomputation stage, for each high level branch and control flow in the code (e.g. packet received but discarded at MAC, AODV timeout and cache flushing and so on) we keep a table of number of instructions executed by target processor. We can keep multiple data bases for different target platforms. At runtime, while executing the code when we come across these branches and control points, we look up from the table to find how many instructions must have executed on target processor.

Here we describe the methodology for creating database for StrongARM SA1100 processor running at 133MHz, used in WINS sensor nodes [9]. We use LARTware [4] cross-compiling tool to cross compile the Qualnet source code into the StrongARM binary format. We use the SimIT toolkit [21] from Princeton, which emulates the strongarm code on linux/x86. We have added a system call into this toolkit that dumps the number of cycles executed so far. We place these system call at all the entry and exit points of the protocol and application code (to ensure that only the protocol code in the simulator is considered and not the simulator



**Figure 4: Performance gain with optimized code**

code). To calculate the time spent in executing the code, we can divide the number of cycles with the processor clock frequency.

Once we can find the duration for which processor is active when servicing a packet, we can use it to model the processing delay. Also from JouleTrack [23] results we know that average current drawn per instruction for this processor is 190.4mA. We can multiply the duration for which processor is active with current drawn to compute the power consumed by CPU.

### 3.4 Power Consumption Model

In this section we describe mechanism by which power consumption for any number of hardware components can be modeled. For each hardware type (e.g. CPU, radio, sensors) we define a set of states (e.g. processor: active, idle, off; sensor: on, off etc) and keep a table that maps the state of device with the current it draws in that state. There could be multiple hardware instances of same type, for example multiple radios in a node; in which case we keep multiple entries but pointing to same table. Any change induced by environment, communication or internal events may result in change of states. The total current drawn is the sum of currents of all components, and this value is fed into the battery model.

### 3.5 Wireless Protocol Stack And Applications

**Wireless Protocol:** We are in process of implementing SMAC and diffusion protocols. For purposes of our experiments we have used MAC802.11 and IP implementation.

**Sensor Applications:** We propose a traffic model that describes sensor network traffic more accurately and an application based on it. We start with observing the fact the most common deployment scenario of sensor network is monitoring or surveillance within a locality. If a phenomenon of interest occurs then all sensors in the vicinity would sense it and become source to relay the information. The second observation is that the destination is also not randomly selected. The purpose of sensor networks is to relay the information to the monitoring stations. That is, the destination nodes are not some randomly selected nodes in the network but the monitoring stations themselves or gateway sensor nodes that connect to these stations.

We can abstract out the above two observations to define a traffic model for sensor networks:

1. The source is a group of nodes that are close to each other. The size of the group depends on the density of the sensor nodes and area in which the sensing effect is observed.

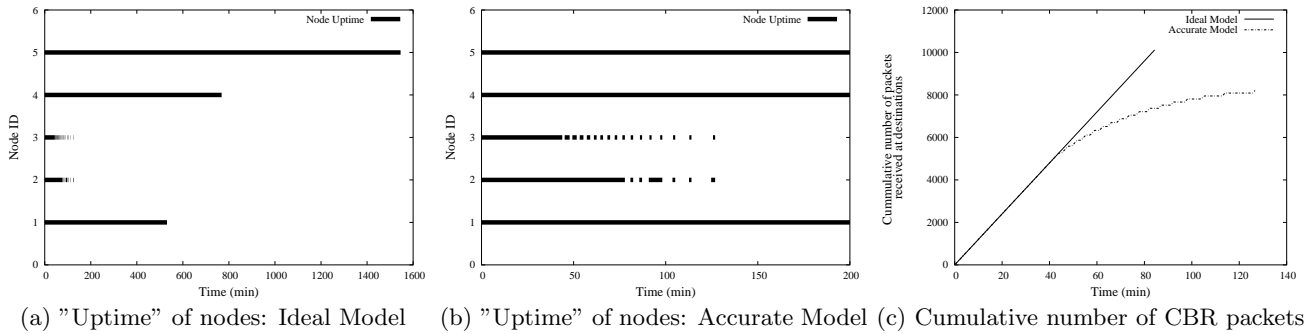


Figure 5: Network uptime and packet delivery for Chain Topology

2. All source in a group connect to same destination. Different groups can connect to different stations, for load balancing or supporting multiple applications. The destinations themselves are chosen from a small subset of nodes.

We describe an application that follows the traffic model described above and it utilizes most of the models that have been presented in this section.

*Monitoring Application:* The sensor nodes are equipped with chemical (toxins) sensors that than detect presence of harmful chemicals in air. Randomly at any given point in terrain an outbreak of chemical occurs for some random duration and some random spread, that is, area it will influence. Any sensor that lies within the spread area of given outbreak, detects it and starts transmitting information to any of the destination node. The information is transmitted with  $K$  packets per second. The parameters to control are density of sensor, number of outbreaks, radius of spread, destination node set and data rate.

In section 4.3 we will see the impact of this traffic model on the relative performance of AODV and DSR routing protocols. It is our hope that this traffic model is used for evaluating protocols and applications in context of sensor networks, rather than simplistic models like CBR that can predict inaccurate results.

## 4. IMPACT OF HIGH FIDELITY MODELS

In this section we consider the effect of using accurate models described in last section on predicted performance of protocols and network. We will see that many measurable quantities in simulation that are relevant to protocol designers and network engineers, are highly dependent on the accuracy of models selected. We show the effect of using accurate battery (Sec 4.1), processor (Sec 4.2) and application model (Sec 4.3) on the network properties as throughput, routing overhead, packet delivery and lifetime is sufficient to stress the necessity in using accurate models for simulations.

### 4.1 Effect of Accurate Battery Model

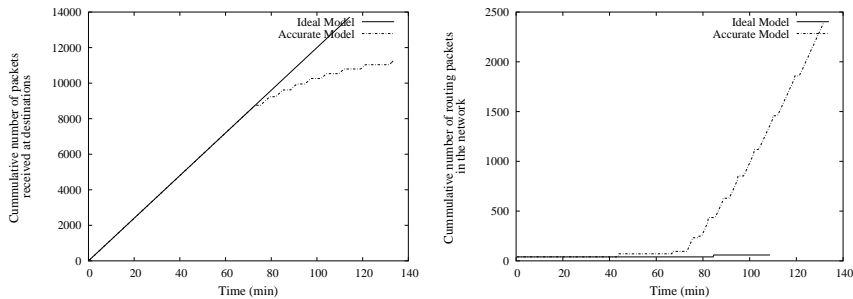
In this section we study the transience phase of network when the nodes in the network are beginning to run out of battery. This unsteady state of network is important to study as it allows to consider the graceful degradation of protocols and networks. We show through the use of the accurate battery model that the network in its final stage of lifetime show fluctuating behavior in availability of nodes and routes. This has direct influence on packet throughput and routing overhead. With an ideal model we observe that

the throughput falls down to zero abruptly, as opposed to gradually decreasing in case of accurate models. This discontinuous behavior of throughput change is unrealistic and difficult for modeling and planning in operational research problems. We also show that the behavior of routing protocols is highly sensitive to details of battery model used. Utilizing accurate model we show that nodes spend a greater proportion of power on routing packet when the network is in unsteady state. This is an undesirable behavior and cannot be observed with ideal models.

We simulated a topology of sensor nodes similar to WINS [9] sensor node. The sensor node is equipped with 100kbps radio. The current drawn from the battery in transmission, reception and idle states are 600, 400 and 20 mA respectively. We assume 802.11 as MAC protocol and IP/AODV for network layer. The parameters for battery are taken from [22]. To focus on the transient phase of network, we consider the case when each node is pre-discharged by 70%. If the battery is discharged prematurely due to high load and it later recovers its capacity, we switch the node "on" if the recovered capacity is greater than 5%. The simulation is carried out, until *all the nodes dies*.

**Chain Topology:** Nodes are arranged in chain of length 5. CBR session is initiated from one end of chain to other, with packet size = 512B and rate 2 packets/sec. As is expected the middle nodes die first (as they spend more energy overhearing communication between other nodes). The network lifetime is when the first node dies. Figs 5(a) and 5(b) shows the uptime of each node when using ideal and accurate battery model, respectively. Observe that in case of accurate model, the nodes do not move from alive to dead state abruptly but alternates between period of available and "down" before finally running out of battery. This is an important result which shows that the behavior of nodes, and therefore network, goes through periods of "up" and "down" before dying.

Fig 5(c) shows the cumulative number of packets received at destination with time. It can be viewed as a plot of packet counter number, which is incremented after successful reception of each packet vs. time. The slope of the curve is the throughput at the destination. In case of ideal battery model, the throughput is constant and remains so until some intermediate node dies and it falls to zero immediately. With accurate model, the slope of the curve decreases gradually and reaches zero. We also note that the network is alive for longer duration, but the total number of packets received at the destination is lower in case of accurate battery model. This discontinuous behavior of throughput drop when using ideal model is a relevant distinction with accurate model.



(a) Cumulative distribution of packets received at the destination (b) Cumulative distribution of routing packets in the network

**Figure 6: Packet delivery and routing overhead for Grid topology**

**Grid Topology:** 16 nodes are arranged in a square grid, where the grid length is set to allow communication between vertical and horizontal neighbors, but not with diagonal neighbors. A CBR session is initiated between two nodes along the diagonal. Fig 6(a) shows the distribution of packets received at the destination, which confirms again that using accurate model avoids the discontinuity in throughput.

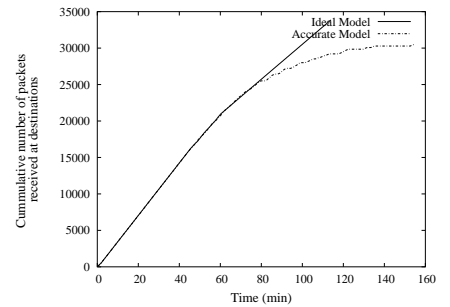
For this topology we show the sensitivity of routing overhead on battery model. Fig 6(b) shows the cumulative number of routing packets in the network with time. It is a plot of cumulative number of routing packets dispatched by all nodes till the given instant in time. For ideal case, we observe that there are two jumps in the number of routing packets send. The first is in the beginning when the source discovered initial route, and second when this route failed. The curve is flat elsewhere. For the accurate model, the plot alternates between the rising and steady states. Also the number of routing packets is almost 100 times than with ideal model. We observe a high sensitivity on routing overhead, and observe that as the nodes show transient behavior in availability, the nodes send more routing packets thus wasting the little remaining life on routing packets. This is an important observation observed only by the use of accurate battery model.

**Random Topology:** Lets apply the lessons we have learned on a larger topology of 50 nodes placed randomly in 1000mx1000m terrain. The cumulative packets received at the destination is shown in fig 7 and cumulative number of routing packets in the network is shown in fig 8. The data packets curve is similar to the one seen before and is a signature for a network that goes through fluctuating states of availability and unavailability. The routing curve is even more interesting. We see that for some duration the overhead with accurate model is lower, but on the whole this overhead is always higher than when using ideal model. The routing behavior is sensitive to the battery model.

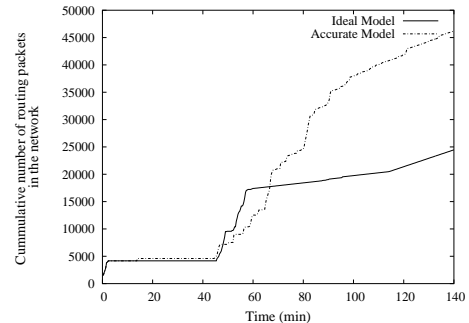
*Summary:* We have seen some interesting properties in network in its transience state through the use of accurate battery model. We have seen that the network is available in fluctuation, the throughput shows discontinuous behavior and routing overhead is highly dependent on battery model.

## 4.2 Effect of CPU power consumption

In this section we emphasize the importance of modeling the processor power consumption. We argue that the percentage of total power consumed in a node by processor is substantial in context of sensor networks and is not



**Figure 7: Cumulative distribution of packets received at the destination**



**Figure 8: Cumulative distribution of routing packets in the network**

a constant overhead that can be easily modeled. This overhead is state and context dependent and requires knowledge of runtime behavior of protocol and application code. The knowledge of power consumed by nodes within a network is important for network provisioner to avoid hot-spots or by protocol designer to compare performance with alternatives. We claim that ignoring this component can predict incorrect trends in power consumption in network or even inversion of relative consumption if we consider only radio (transmission and reception) power consumption.

The power consumed by processor depends on what action it takes upon any event (reception of packet, timer trigger etc), which in turn depends on state of the protocols and context in which the event occurs. For example, consider the case when a unicast packet destined for node 1 is received by nodes 1 and 2. Node 2 would drop this packet right away at the MAC layer, while the first node will process it and possibly send to network layer also, thus executing more code than node 2. The radio power consumption would predict equal power consumed by both nodes, however, if we account processor cost then node 1 consumed more power than the other. With this observation we find that any study of power consumed in network cannot be predicted accurately by considering radio power consumption only.

To show the above concept in action we simulate a square grid topology of 49 nodes. The radio is a 802.11b running at 1Mbps. The current drawn for transmission and reception is 600mA and 400mA respectively. We have simulated the StrongARM SA1100 processor running at 133MHz. The current drawn per instruction is 190.4mA as reported in JouleTrack datasheet [3]. Three CBR sessions are initiated between random source and destination pairs. The routes

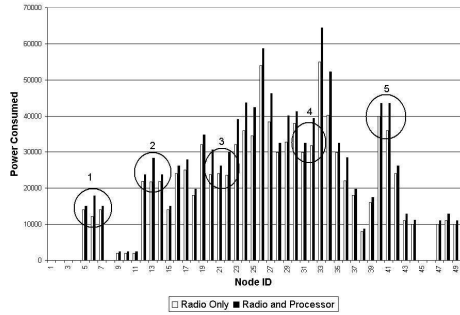


Figure 9: Power Consumption by nodes in the network.

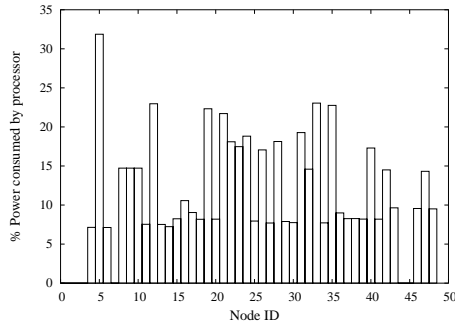


Figure 10: % Consumption by processor

are pre-computed to ignore the routing overhead in our measurements. Fig 9 gives the energy consumed by each node. The white bars show the radio energy consumption (referred to as simple model), while the black bar denotes the total energy consumption of both radio and processor (the detailed model). Fig 10 gives the relative percentage of power consumed by processor for each node.

We have highlighted five places in the graph that shows the difference between the simple and detailed model. These are discussed in order shown in graph.

1. The energy consumed as reported by simple model for node 6 is less than nodes 5 and 7. However, in detailed model the situation is reversed. Thus it is dangerous to make assumption on relative ordering of power consumption in node based solely on radio power model.
2. In the second case, the simple model predicts the power consumption for the three nodes as equal. However, with detailed analysis we see that the middle node is actually consuming more power. Thus there can be hotspots in network which are visible only by using detailed model.
3. This is similar to case two, except the middle node is consuming less power.
4. This case demonstrates that the relative power consumption also varies if we take into account the processing consumption. The second node consumes 6% more power when using simple model, but 21% when using detailed model.
5. And finally, a reversal effect can happen when the simple model predicts different values but the detailed model predicts equal. That is, a hot-spot predicted by radio model can actually be a false positive.

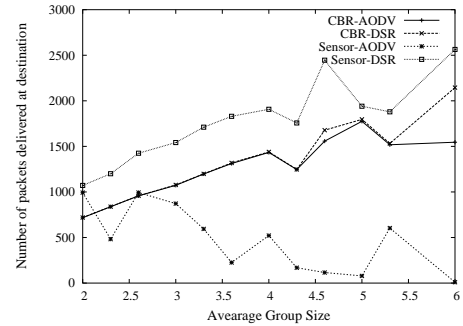


Figure 11: Relative performance of AODV and DSR routing protocol under two traffic models.

Given any topology and traffic pattern, we observe that these effects are always visible which strengthens our claim on the need to model processor cost in node power consumption.

### 4.3 Performance under sensor traffic model

To evaluate relative performance of MANET routing protocols the ad-hoc network specific random source-pair destination traffic model is used and this trend has followed into the sensor network. We, however, claim the need for more realistic traffic model to compare protocols for sensor networks. We compare performance of AODV and DSR routing protocols using MANET based and sensor network based traffic model. We observed that the relative performance of these protocols are highly sensitive to the underlying traffic model used.

A 100 node square grid topology of sensor nodes is simulated. We have used the traffic model described in sec 3.5. In each run of simulation, three *outbreaks* of duration 3 minutes each are simulated. Each outbreak has varying radius of *spread* such that the average size of group varies from 2 to 6. Each connection between source and destination is CBR traffic of 256B packets at rate of 1 packet/sec. The network is simulated for period of 15 minutes. The destinations are the four corner nodes of the grid, and for a particular outbreak a destination is chosen randomly from this set.

In order to evaluate this traffic model against a random source-destination pair CBR model, we define an equivalent CBR traffic for each run of simulation. The equivalent traffic is defined as having (number of groups)\*(average size of groups) of flow between random pair of source and destinations with duration 3 minutes each and same packet and rate as above. Thus, the offered load is same in both models only the traffic pattern is different. The number of packets delivered by both protocols using these two models is shown in fig 11. It is observed that in CBR traffic model, both AODV and DSR perform equally well, which is expected since the offered load is less than maximum throughput of the network. However, when we consider sensor traffic model, we find some surprising results. AODV performs very poorly and its performance degrades with increasing group size. The DSR protocol, however, performance ever better than CBR traffic model and its throughput increases with increasing group size, which means that it has not yet reached the saturation point.

This experiment shows how the relative performance of routing performance is closely related with the traffic model used. Between random source-destination pair model and

group-source and fixed-destination models, we have seen that the relative performance of routing protocol varies significantly. It is our hope that more realistic traffic models, that represent sensor network specific traffic patterns, should be used when evaluating protocols.

## 5. CONCLUSIONS

In this paper we have presented the suite of models to facilitate simulation of sensor networks in a detailed fashion. The suite of models include sensing stack and sensing channels that model wave-propagation and diffusion based sensing phenomenon; an accurate battery model with our implementation optimizations to enable its use in network simulations; model for accounting the delay and power consumed by processor in executing protocol and application code; sensor hardware specific power consumption model and a traffic model.

We have also investigated into the effects of using detailed models on the network level performance. We have seen that using an accurate battery model has influence on network lifetime, its availability, the throughput at the destinations and routing overhead. On accounting the processing power consumption we found trends which are incorrectly represented if we consider power consumption from radio only. And finally, we found that the relative performance of routing protocols is highly related to the traffic model used. For the same offered load in network, AODV and DSR routing protocols perform equally well with random source-destination pair based traffic model. However, with the sensor traffic model DSR outperforms the AODV protocol.

Due to high sensitivity of network and application performance statistics on the details on the simulation models used, we would like to stress the importance of using accurate models for meaningful simulations of sensor networks.

## 6. REFERENCES

- [1] Atemu. <http://www.cshcn.umd.edu/research/atemu/>.
- [2] Fortran programs for simulation of electrochemical systems. <http://www.cchem.berkeley.edu/~jsngrp/fortran.html>.
- [3] Jouletrack - a web based tool for software energy profiling. SA1100 models. [http://www.mtl.mit.edu/research/icsystems/uamps/pubs/files/sinha\\_armInstEnergy.pdf](http://www.mtl.mit.edu/research/icsystems/uamps/pubs/files/sinha_armInstEnergy.pdf).
- [4] Lartware. <http://www.lart.tudelft.nl/lartware/compile-tools>.
- [5] Qualnet. <http://www.scalable-networks.com>.
- [6] Sense. <http://www.cs.rpi.edu/~cheng3/sense>
- [7] Simple scalar llc. <http://www.simplescalar.com/>.
- [8] Mica notes. <http://www.xbow.com/>.
- [9] J. R. Agre, L. P. Clare, G. J. Pottie, and N. P. Romanova. Development platform for self-organizing wireless sensor networks. *Proc. SPIE, Unattended Ground Sensor Technologies and Applications*, Vol 3713.
- [10] M. Doyle and J. Newman. Comparison of modeling prediction with experimental data from plastic lithium ion cell. *J. ElectroChem. Soc.*, 143(6), 1996.
- [11] J. Heidemann, N. Bulusu, J. Elson, C. Intanagonwiwat, K. chan Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan. Effects of detail in wireless network simulation. *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference*, September 2000.
- [12] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. *International Conference on Mobile Computing and Networking (MobiCOM)*, August 2000.
- [13] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 2003.
- [14] D. Linden and T. B. Reddy. *Handbook of Batteries*. Third edition.
- [15] J. Liu, D. Nicol, F. Perrone, M. Liljenstam, C. Elliot, and D. Pearson. Simulation modeling of large-scale ad-hoc sensor networks. *European Interoperability Workshop 2001*, June 2001.
- [16] J. Liu and D. M. Nicol. Dassf 3.1 user's manual. <http://www.cs.dartmouth.edu/~jasonliu/projects/ssf/papers/dassf-manual-3.1.ps>.
- [17] Ns-2. <http://www.isi.edu/nsnam/ns>.
- [18] D. Panigrahy, C. Chiasserini, S. Dey, R. Rao, and A. Raghunathan. Battery life estimation of mobile embedded systems. *Proc. VLSI Design*, 2001.
- [19] S. Park, A. Savvides, and M. B. Srivastava. Simulating networks of wireless sensors. *Winter Simulation Conference*, 2001.
- [20] L. F. Perrone and D. M. Nicol. A scalable simulator for tinyos applications. *Winter Simulation Conference*, 2002.
- [21] W. Qin and S. Malik. Automated synthesis of efficient binary decoders for retargetable software toolkits. *Design Automation Conference*, June 2003.
- [22] D. Rakhmatov and S. B. K. Vrudhula. Energy management for battery-powered embedded systems. *ACM Transactions on Embedded Computing Systems*, 2002.
- [23] A. Sinha and A. P. Chandrakasan. Jouletrack - a web based tool for software energy profiling. *Design Automation Conference*, June 2001.
- [24] Sundresh, S. W. Kim, and G. Agha. Sens: a sensor, environment and network simulator. *Simulation Symposium*, 37th Annual:221– 228, April 2004.
- [25] M. Takai, J. Martin, and R. Bagrodia. Effects of wireless physical layer modeling in mobile ad hoc networks. *International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, October 2001.
- [26] C. Ulmer. Wireless sensor probe networks - sensorsimii. <http://www.craigulmer.com/research/sensorsimii>.
- [27] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. *INFOCOM*, June 2002.
- [28] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin. The design and use of simplepower: A cycle-accurate energy estimation tool. *Design Automation Conference*, 2000.
- [29] X. Zeng, R. Bagrodia, and M. Gerla. Glomosim: a library for parallel simulation of large-scale wireless networks. *Parallel and Distributed Simulations (PADS)*, May 1998.