Simulation Scalability Issues in Wireless Sensor Networks

E. Egea-Lopez, J. Vales-Alonso, A. S. Martinez-Sala, P. Pavon-Mariño, J. Garcia-Haro[‡]
Department of Information Technologies and Communications
Polytechnic University of Cartagena, Spain
e-mail: {esteban.egea, javier.vales, alejandros.martinez, pablo.pavon, joang.haro}@upct.es
[‡] Corresponding author. Address: Campus Muralla del Mar, 30202, Cartagena, Spain
phone: +34 968 325314, fax: +34 968 325973, e-mail: joang.haro@upct.es

Abstract— **The formidable growth of WSN research has opened challenging issues about their performance evaluation. Despite the steady increase in mathematical analysis and experimental deployments, most of the community has chosen simulation for their study. Although it seems straightforward, this approach becomes a quite delicate matter. Complexity is caused by several issues. First, the large number of nodes heavily impacts simulation performance and scalability. Second, credible results demand an accurate characterization of the sensor radio channel. New aspects, inherent in WSN, must be included in simulators, e.g. a physical environment and an energy model, leading to different degrees of accuracy versus performance. Moreover, many necessary models are in the continuous-time domain (e.g. heat transmission, battery discharge), being complex to integrate into discrete event network simulators. These issues result in an exponential growth of the overall network state information. Through this survey we review these problems both quantitatively and qualitatively while depicting a common suitable simulation model. We also briefly describe the most significant simulation frameworks available.**

Index Terms— **Wireless sensor networks, simulation, modeling, discrete-event simulators.**

I. Introduction

In recent years extensive research has been conducted on Wireless Sensor Networks (WSN), being considered one of the top research topics. Essentially, sensor nets face technical problems similar to those of Mobile Ad-hoc NETworks (MANETs). However, WSN are formed by a large number of resource-constrained and inexpensive nodes, which has an impact on protocol design and network scalability. Energy is a primary concern, because nodes usually run on non-rechargeable batteries. Thus, the improvement of network lifetime is a fundamental research issue. In addition, the envisioned applications for WSNs and the operation of the protocol layers are driven by physical sensor measurements, rather than voice or user-data services.

It is almost unfeasible to analytically model a WSN and predict the actual performance of high-level protocols and network operation. Deploying field test-beds to study the actual behavior of protocols implies a great effort. Moreover, the first real WSN applications are currently being explored [1] and many are yet to come.

Consequently, simulation is essential in the study of WSN, being the common way to test new applications and protocols in the field. Indeed, there has been a recent *boom* of specific simulation tools for WSN modeling [2]. Many of these new tools have been designed with goals different from those of the "classical" network simulators, bringing new approaches to network simulation. This renewed interest in simulation contrasts with several concerns over simulation practice.

On one hand, there exists an increasing concern about the methodology [3] and assumptions used in simulation of *wireless* networks [4], [5]: idealized hardware, simplified protocols and non-realistic radio models too often lead to mistaken results. Indeed, experiments [1] warn about subtle effects which should be taken into account. For instance, batteries do not supply current linearly, which affects sensitivity and transmission power and eventually protocol performance. However, including the required degree of detail usually adds hard computational requirements, which compromises simulation scalability. On the other hand, the limits of scalability of network simulation are being questioned [6]. The additional burden of propagation computation and integration of continuous-domain models may force a reduction in the size of the simulated network. Whereas wired network simulation seems to have overcome past doubts regarding scalability, wireless network simulation faces renewed challenges. To sum up, the classical simulation tradeoff stated as "accuracy and detail versus performance and scalability" reappears in WSN simulation with increased strength and even more ambitious requirements. Namely,

- WSN simulators must be provided with highly accurate radio models that in addition scale well to a large number of nodes.
- New model components (sensor hardware, batteries, CPU model) and a tight cross-layer coupling, not considered in classical tools, must be included.
- A synthetic characterization of the environment under study is needed, in order to include the dynamics of the physical parameters in the analysis and evaluation of protocols. However, integrating physical continuous magnitudes (e.g. heat transmission) into discrete event simulation may increase considerably the model complexity.

The aim of this article is to provide insight into such issues and to describe the building blocks of a general simulation model for sensor networks, introducing their specific problems and current challenges. We first introduce a general simulator architecture. Then, we discuss and review open problems and proposed solutions to technical issues of the different components. To provide a quantitative comparison of their scalability, we have conducted a set of simulations, measuring the effect on performance of the key WSN model components. Next we briefly review the differences between generic and WSN-specific simulators. Finally, the main open research issues and conclusions around this work are presented.

II. A MODEL FOR WSN SIMULATION

At a network-wide scale, WSN are composed of a (very) large number of sensors or *nodes*, which gather events and process them. Some WSNs, and simulation tools, also include *sink* nodes. They process data from the net, and may interrogate sensors about events of interest. The events come from the physical *environment* component, which may be generated by itself, or triggered by *agents*.

Due to the hard constraints of sensors, the classical layered approach is not suitable. Node behavior depends on interacting factors that cause cross-layer interdependencies. A convenient way to describe it is to divide nodes into abstract tiers, as represented in Figure 1. Namely,

- The *protocol-tier* comprises all the communication protocols. Their operation usually depends on the state of the physical tier.
- The *physical-node* tier represents the underlying hardware and measurement devices.
- The *media-tier* links the node to the *"real world"* through (1) a radio channel and (2) one or more physical channels, connected to the environment component.

In this architecture, layers need to exchange information that would be isolated in the traditional OSI model. This tight coupling affects the simulation architecture in different ways:

- The overall design must provide an efficient mechanism to share information between modules, without degrading performance.
- The interface between components must be flexible and extensible. It is not clear *when* and *who* needs information. Fixed interfaces and primitives between layers or components should not be assumed. For instance, an estimate of link state may be used by MAC, routing and application layers.

The publisher/subscriber software paradigm accomplishes the previous functions. Entities publish their available information and subscribe to others, so that they are informed on changes. It may be implemented as a global "blackboard" where each component can write down changes on its information to be read by others. This "blackboard" is used, for instance, by the OMNET++ Mobility Framework (Table II). NS-2, on the contrary, uses the object-oriented approach of public methods, prone to code scrambling. A design based on components, such as the one by J-Sim, is also appropriate because a clear interface (*contract*) must always be declared, which, in turn, is easily extensible.

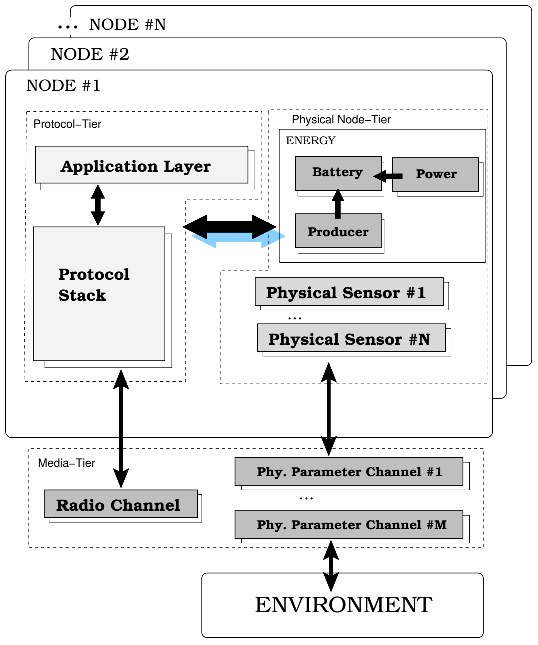


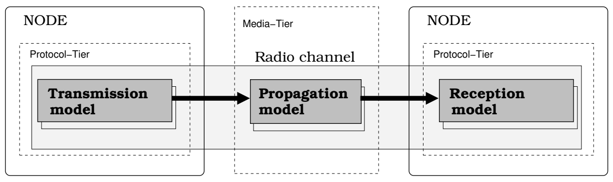
Fig. 1. Network-level WSN model

Every simulated layer increases the processing power needed and the simulation run time. If available, substituting a layer by a mathematical abstraction may alleviate this problem. The layer becomes then a computationally efficient black box, interacting with the rest of the modules. There are a number of papers that derive analytical representations of communication layers, including the cross layer dependencies of WSN [7].

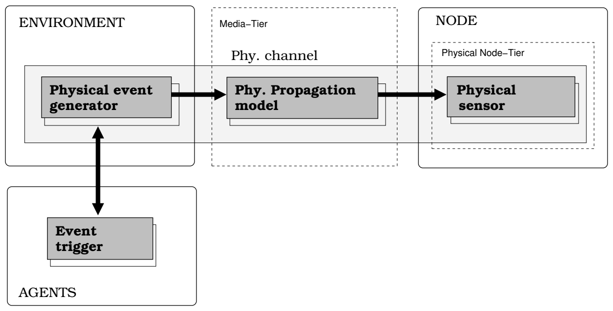
A. Radio channel architecture

Figure 2.(a) depicts the relationship between the radio channel and the former tier-based model. Synthesized radio channels must determine (1) the nodes that receive a transmission, (2) the quality of the reception (with or without error) and, (3) the state of the shared medium (busy or free). To implement such functionality, most simulators employ three independent modules:

1) The *transmission module* defines the radiated power, frequency, data rate and other transmission parameters.
2) The *propagation module* computes the received power, which is mainly a function of the transmission parameters and the distance. The propagation model used can be deterministic (e.g. free-space, two-ray ground reflection) or add some random component (e.g. shadowing). Besides, a correct selection of model parameters according to the network particular application is mandatory. Indeed, scope-matched models (i.e., indoor, grass) derived from empirical studies [8] should be considered if available.
3) Based on the received power and on the internal transceiver operation (modulator, sensitivity) the *reception module* decides whetherpackets are received, whether there is an error, or whether the medium is busy. In addition, this module decides how to treat interference.



(a) Radio



(b) Environment

Fig. 2. Comprehensive models

B. *Physical environment architecture*

Figure 2.(b) describes the structure of the environment model. Sensors are fed with data from the environment through *physical channels*. These channels are in charge of deciding when and which nodes receive the physical events generated by the *physical event generator*. For instance, consider a WSN monitoring temperatures, in this case the event generator may consist of heat sources activated at different points, and the physical channel must implement the physics of the heat transmission to compute the temperature received by each node at a given instant.

In addition, some WSN simulators incorporate independent *agents* (e.g. a mobile vehicle) that trigger events in the physical event generator.

C. *Energy module architecture*

For an accurate description, consumption should be controlled by means of two different modules (see Fig. 1): the *power module*, which computes the power consumption of the different components, and the *battery module*, which uses this information to compute the battery discharge. Indeed, it is *not* the power but the battery model which is responsible for checking if the node has exhausted its battery.

Besides, energy-producers inside the nodes may be considered, for instance to model solar or even wind powered sensors. These components are introduced via *producer* modules connected to the battery module.

III. Radio channel modeling issues

Studies on MANETs and WSNs have intensified the need for detailed radio models in simulators. Selecting which stations receive packets is no longer obvious. Wireless simulators require computation of the propagation for every possible receptor. Moreover, packets suffer degradation caused by concurrent transmissions (interference), which are also highly related to the underlying channel model. Consequently, simulation results can be very different depending on the chosen radio model. On one hand, simple models do not match real behavior and may lead to misleading conclusions [4]. On the other hand, complex ones can achieve a high fidelity, but they are computing and memory intensive and its implementation is not straightforward. Indeed, propagation computation is one of the most limiting factors of wireless simulators. The goal is to obtain accurate enough models that can be seamlessly scaled to several thousands of nodes.

Scalability is limited by the computation of propagation and interference. Let us explain the relationship with the limiting resources: processing power and memory consumption. Every packet transmission requires *for every node that is affected by the transmission* the computation of the received power, which consumes processor cyles, and the scheduling of a reception event, which consumes memory. Thus, the decission on "which nodes are affected" has a deep impact on both resources. However, the real scalability problem is a matter of *the number of times* that computation of these propagation losses is performed. These calculations are unavoidable when:

- The nodes move over time, and so does the received power.
- The channel is stochastic, i.e. the propagation loss is a random process that changes over time.

The location of WSN nodes is usually fixed and, hence, computation of propagation prior to simulation is possible. Then, it becomes a matter of fast searching in a table (section IV). Still, a rich stochastic modeling of the channel is worth being included. Therefore, such accurate models ought to recalculate the power received *at every packet transmission*, or, at least, as a function of the time scale of channel variation, if it is known. The memory consumption is a function of the number of packet reception events that must be scheduled, which depends on how interference is modeled.

Whenever a new transmission starts, it may affect simultaneous transmissions in progress. Its influence on concurrent transmissions depends on the accumulated power of interfering signals, since packet loss probability is a function of the Signal to Interference and Noise Ratio (SINR). Thus, for each packet transmission, the nodes to which the new transmission must be notified have to be selected. A reception event must be scheduled for them in this case.

Three models are commonly considered to compute the effect of interference and propagation. Namely,

1) *Disk model and collisions*. A reception event is scheduled only if the received power is above a given threshold (or equivalently, a transmission range). Any concurrent transmission results in a collision. When the propagation model is deterministic and no mobility is considered, no propagation recomputation is necessary after an initial calculation. Otherwise, the transmission range and/or the affected nodes must be recomputed. The node density determines the memory consumption.

For instance, considering a 1,000 node network, if an average 5% of them are in range of each other, only 50 reception events are scheduled per packet transmission. This assumption -collisions- overestimates the interference level for in-range but distant nodes, i.e. the capture effect is neglected. Moreover, this approach also underestimates interference and channel occupation for nodes beyond range.

2) *Limited interference*. Interference is considered if nodes are within a certain range (carrier sense threshold). In this case, depending on the SINR, there may occur a collision or a capture effect. As in the previous case, the noise level beyond range is underestimated. The number of nodes involved in computations is the same as in the previous numerical example. This is the most usual approach, because it may significantly reduce the set of nodes to compute propagation for a given transmission, while providing a reasonably accurate model.
3) *Full interference model*. The received power of every transmission in the network (no matter how faint) is used for SINR evaluation. Two refinements are obtained: first, interference increases noise level and may spoil a reception (SINR evaluation), and second, the channel is sensed busy more frequently. This model forces to schedule one event in every node for every data packet sent. For instance, in the conditions of the previous example, 1,000 events must be scheduled for every packet transmission. Obviously, this strategy consumes a huge amount of memory. Mobility and/or stochastic propagation models require also recomputation of the received power.

IV. Radio channel optimizations

Simulation tools currently explore various solutions to enhance the performance of the radio channel. Some insight into these methods is provided in the following sections.

A. Partitioning procedures

When a disk or limited interference model is used, a smart implementation may reduce their impact on scalability.

Partitioning procedures do not seek to reduce the number of propagation recomputations, which is imposed by the accuracy of the model, but the number of calculations performed in each recomputation. The strategy is to rapidly filter the nodes for which received power is computed and events scheduled. Indeed, great savings can be obtained if the simulation area is relatively large and the node density is low, since the majority of the nodes are out of range, and it is not necessary to schedule events for them. For instance, in Fig. 3.(a) a random set of nodes is shown. Assume the transmission range is set fixed and equal to R. If no partitioning is used, the propagation is computed for every possible pair of nodes (e.g. 1-2, 1-3, 1-4, 1-5, 1-8, etc.), even though for several pairs (e.g. 1-5, 1-8, etc.) it is not necessary.

Naoumov *et al.* [9] propose two partitioning procedures:

1) To divide the simulation area into a grid of cells. Propagation is only computed for nodes belonging to cells around the transmitter. The improvement is sensitive to the definition of cell and its size, and to the number of location updates, that is, the mobility rate. For instance, in the example of Fig. 3.(b), the simulation area is divided into a grid of 4x5. If propagation is computed only in adjacent cells, the computations for node 1 only include the pairs 1-1 to 1-5. Although pair 1-3 and 1-5 calculations are still superfluous, there is a remarkable reduction in the propagations computed.
2) To create a double-linked list of all the nodes sorted by their X-coordinates, providing a fast search of nodes that fulfill $X_{destination} \in [X_{source} - R, X_{source} + R]$. The process also filters the list by the Y-coordinate. Propagation is computed for nodes matching both conditions. Fig. 3.(c) exemplifies this procedure. First, nodes within area A are selected. Then, nodes within $A \cap B$ are selected, and the propagation losses are computed for them. Like the grid optimization, the computation of some pairs (1-3 and 1-5 in the example) is not necessary, but the global computation saving is considerable.

The problem of partitioning is that it cannot be used if interference is fully modeled and it is sensitive to the density of nodes and mobility, which requires to update the data structures (grid or list).

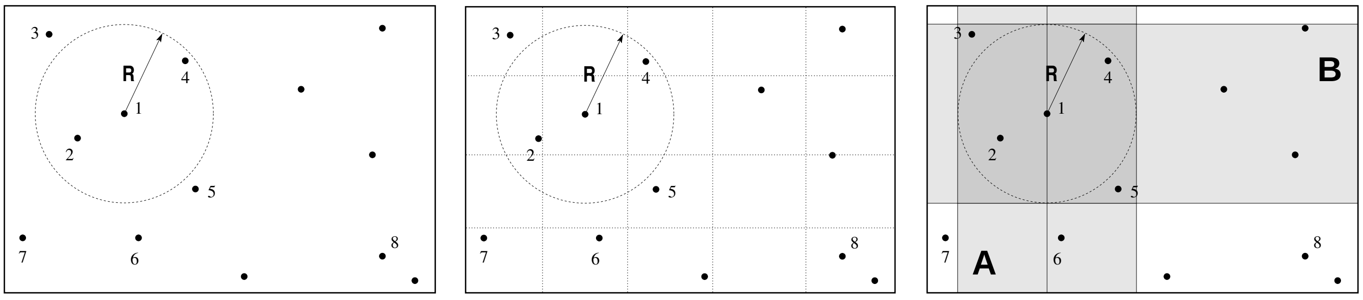
B. Parallelism

Another option, supported by some simulation frameworks, is to parallelize computations. This type of optimization may be used for any interference model considered. Parallel simulators divide the network into a number of partitions, running simultaneously in different processors. For instance, consider the example of Fig. 3.(b) in a bi-processor machine. Two partitions may be established (e.g the two topmost and the two bottonmost rows). Each processor executes the computations of propagation for its corresponding partition independently.

Transmitted messages are delivered to neighboring partitions, which compute propagation, and so on. Even so, the utility and future of Parallel And Distributed Simulation (PADS) has been discussed for years [10]. The complexity of developing parallel simulations and the cost of the equipment needed are the factors that prevent a widespread PADS usage. These factors together with the need for a quick test of novel networking techniques make sequential simulation remain the norm.

C. Connectivity graphs

Another solution is the generation of connectivity graphs to avoid propagation computation. The idea is to associate probabilistic communication properties, derived from empirical data, to each link. That is, to substitute the propagation calculations by a mathematical abstraction of the link layer that provides a packet loss probability or any other property of interest. Indeed, these properties are independent for each link. The result is a -*connectivity*- graph. Cerpa *et al.* [11] describe several methods to obtain and validate such network graphs. A more accurate characterization of radio links can be obtained at a reduced computational cost. Specific WSN



(a) Without optimizations (b) Grid optimization (c) Lists optimization

Fig. 3. Propagation computation and optimizations

simulators as TOSSIM (Table III) already use experimentally derived connectivity graphs.

Interferences can be modeled as collisions, or by deriving a reception probability function. For instance, in [12] a simple model is proposed: let p_{ij} the probability of succesful reception for a transmission from node i to node j. The probability of succesful reception when node a transmits to b in the presence of k concurrent transmitters equals $p_{ab} \prod_{i \in k} (1 - p_{ib})$.

Even then, the most important drawback of this technique is the lack of available empirical data sets and the great effort needed to collect them.

V. Environment modeling

A key component of a WSN simulator is the environment but its modeling is often neglected. Recent works like [13] have emphasized this deficiency and introduced some alternatives. Basically, three aspects should be addressed:

1) Which data are fed to the nodes?
2) Which nodes receive these data?
3) How do nodes interact with these data?

Regarding the first question, the main alternatives are: (1) using a data set, measured in real environments. This data set may be refined by interpolation. And (2) to model the environment utilizing some well-known or approximate mathematical model. The first alternative provides the real characterization of the problem under study. Therefore, these results are more reliable than those from synthesized data. However, there is a lack of available data sets for different environments, which limits the experiments. Mathematical models are useful in this case, though they are usually models in the continuous-time domain. Hence, it is necessary to compute when the physical magnitude is sensed by the nodes and its strength. Simulators use two approaches to deal with this problem.

Modeling the environment as an analogy of a *disk* radio channel is the most common approach. In this case, the reception range of the physical event is computed from a physical model, and the nodes in range are notified. For example, J-Sim (Table II) uses the concept of *SensorChannels* in a similar way to that of radio channel. Through agents (see Fig. 2.(b)), stimuli are fed in the *SensorChannels*, which schedule the events to all the nodes *in range*. J-Sim implements acoustic and seismic wave models, computing the range from their propagation equations. The discussion on propagation scalability given for the radio channel can be equally applied here. The most important property of this mathematical model is that computations are only performed *when the physical event happens*. No computation is needed otherwise.

Many interesting situations cannot be modeled in this way. For instance, the diffusion of many physical magnitudes (e.g. heat transmission), which are described by means of partial differential equations systems. Most of these equations do not have a known solution, and hence, they have to be approximated by numerical methods, like difference equations . These methods require to discretize time, that is, to sample the physical magnitude every Δ time units. Indeed, in practical methods Δ is small, so as to avoid instabilities of the numerical method. Therefore an event to recompute the physical environment status must be scheduled frequently (every Δ time units), which becomes a noticeable time and memory consuming task. The simulation engine must calculate the nodes that receive physical events continuously if one of these models is employed. In fact, this is the real problem of integrating continuous models into discrete-event engines.

Additionally, the presence of *actuators* in the network may be considered. These modify the environment like the agents do, but, in addition, they react upon environmental conditions, triggering new events. For instance, in a thermoregulation application, once a sensor detects a peak of temperature it triggers a cooling device. Thus, networks with actuators require a greater interaction with the physical media. Sensor measurements trigger actuator responses that stimulate the physical channels, which, as a consequence, generate new environmental events. The implementation of this two-way feedback is not straightforward, and it limits scalability due to the burden of new generated events on the overall performance.

To sum up, including realistic environments into WSN simulators is mandatory if we are to achieve reasonable conclusions. However, it constitutes a major scalability issue.

VI. Energy model

Many ongoing studies focus on optimizing the energy consumption by using specialized protocols. There is a general agreement on the need for accurate simulation models for power consumption, in order to reach meaningful conclusions.

As shown in Fig. 1, the *power* module informs the *battery* module of the current power consumption. Based on this data, the latter computes the battery discharge and controls when the node dies. The *power* module may be internally accessed from several components, which report their power consumption based on their particular state. For instance, J-Sim uses a similar approach: the CPU and the radio communicate the value of current that has to be drained from the battery.

Regarding the *battery* module, simple models consider an ideal source discharging linearly. That is, the energy of a source starting with E_0 Joules consuming a power P during t seconds, is given by $E = E_0 - Pt$. When E is equal to 0, the battery is considered exhausted. However, this linear relationship is not an accurate representation of the real physics of a battery. In fact, E depends on the discharge profile [14] of the battery. In other words, E is a non-linear function of the different power consumptions during operation. Such models, continuous in nature, must also be adapted to discrete simulation. Reference [14], e.g., provides an algorithm to integrate an accurate non-linear battery model into the idealized NS-2 energy model. This discretized approximation computes the battery discharge at *each change of state reported by the power module* (e.g. radio or CPU changes of state). Besides, this process must be run independently in every node. Certainly, it requires significant processing resources, since this process is run almost continuously. This is another example of how the integration of a continuous model causes a scalability problem.

VII. Scalability evaluation

To assess the influence of the discussed issues on the simulation performance, we have developed the following experiment: a basic model, developed with OMNET++ (Table II), has been simulated with 125, 500, 1000, 5000 and 10000 nodes. Simulations were run on a P-IV 2.8 GHz processor with 4 GB of RAM memory. This basic model uses an implementation of a WSN MAC protocol (S-MAC) and a static Minimum Spanning Tree routing. A sink node is located in the center of the area. Nodes inform the sink about the occurrence of a physical event. A disk model with collisions is used for radio channel. Propagation losses are computed only at initialization. The environment module is present but no events are generated. A linear battery model is used. Afterwards, the simulations are reproduced replacing one single module of the basic model with another more accurate/complex:

- Environment. A disk model with fixed reception radius. Both time and position of physical events are randomly generated.
- Radio. Full interference model.
- Battery. A non-linear continuous model, using the algorithm proposed in [14].

Finally, all the complex models (environment, radio, battery) have been simulated together. Table I shows the results in terms of simulated seconds per real seconds and average memory consumption. These results lead to the following conclusions:

- Regarding propagation, the full interference model exponentially decreases the performance, with respect to the basic model. Indeed, the ratio of simulated second to real second is below 1 for simulations with more than 1000 nodes. It means that, for instance, simulating 3 days of a 10000-node network requires 500 real days! This confirms the impact of the selected radio model on simulation scalability.
- The commonly used disk model for environment has little influence on the overall performance. But this is the "lightest" model, because it does not require complex computations. More important, it is well-suited for discrete-event simulation. The problem arises when a continuous-domain model is used to describe an environment as the battery model does.
- The battery model decreases performance in two to three orders of magnitude. In order to solve the underlying continuous time equations a high sampling frequency is needed (in this case, every change of state). Besides, the evaluation of battery discharge requires costly operations. Indeed, this is the dominant factor also in the full model. This result clearly shows that integrating a continuous-domain magnitude into a discrete event simulator is a major scalability challenge. In fact, it must be faced because the use of such magnitudes seems unavoidable to achieve accurate WSN simulations.

In summary, our results confirm that large simulations of WSN with a minimal accuracy are not practical with current approaches.

VIII. Simulation frameworks

In this section we briefly review the most interesting capabilities, advantages and drawbacks of the existing tools for WSN simulation. Reference [2] provides a deeper discussion on this topic. The options are: specific add-ons to general communication network simulators (Table 1), and WSN simulation frameworks built from scratch (Table 2).

A. Classical network simulation frameworks

Tools such as DaSSF, JiST, J-Sim, NS-2, NCTUns, OMNET++, and Ptolemy fall into this category. Their main characteristics are summarized in table II.

They use a layered architecture which is similar to the real protocol stack implementations, atop a simulation engine. Additional components describe topology and links and glue everything together. Availability of ready-to-use models, powerful scripting support for complex network instantiation, clear and extensible interfaces and graphical support are some criteria to select them. Most of these have been extended with wireless components such as radio channels, node mobility and energy models. Some also support sensor networking.

Before selecting these tools for simulation, two issues should be considered: (1) their protocol implementation is usually quite different from a real one. Thus, real system deployment of simulated proposals requires changes of implementation. (2) New communication paradigms (*data* or *location-centric*) and highly application-dependent operation

#Nodes	Basic model		+ Environment		+ Interference		+ Battery		Full model	
	simsec/sec	RAM [Mb]	simsec/sec	RAM [Mb]	simsec/sec	RAM [Mb]	simsec/sec	RAM [Mb]	simsec/sec	RAM [Mb]
125	268.6	8.78	162.0	8.9	126.4	9.06	0.5931	12.79	0.1494	13.14
500	30.16	17.77	26.15	17.9	3.594	22.17	0.01724	36.29	0.0162	40.07
1000	12.69	29.76	11.65	29.89	0.7641	44.69	0.008206	78.46	0.00743	81.318
5000	1.834	122.6	1.761	122.7	0.0337	433.1	0.003422	228.6	0.002830	494.373
10000	0.7973	233.7	0.7687	234.8	0.006	1462	0.001755	418.8	0.001306	1595

TABLE I
PERFORMANCE RESULTS FOR DIFFERENT MODELS

	Language/ Scripting	Available modules*	Graphical support	Emulation	Additional notes
†DaSSF/TOSSF	C++/DML	C/W/A/WSN+	Proprietary	Limited	Designed for parallel simulation. TOSSF is a WSN extension that simulates native TinyOS code. *http://www.ssfnet.org*
JiST/SWANS	Java/Jython	W/A	–	Yes	Provides real Java application execution on a simulated network. Ad-hoc network support. *http://jist.ece.cornell.edu*
J-Sim	Java/Jacl	C/W/A/WSN+	Good edition and debug	Yes	Component-based design allows for easy composition. Very detailed sensor extension. *http://www.j-sim.org*
NCTUns	C	C/W/A	Excellent edition and debug	Yes	Uses a modified UNIX kernel to simulate, supporting the use of the real TCP/IP stack and real applications. Ad-hoc network support. *http://nsl.csie.nctu.edu.tw/nctuns.html*
†NS-2	C++/OTcl	C/W/A/WSN	–	Limited	The most used. Extensive library of protocols, including WSN proposals. *http://www.isi.edu/nsnam/ns*
†OMNET++	C++/NED	C/W/A	Limited for edition, excellent for debug and runtime animation	Limited	Powerful GUI. Its mobility framework may be used for sensor simulation. *http://www.omnetpp.org*
Ptolemy II	Java	C/W/A/WSN+	Excellent edition and debug	Yes	Concurrent simulation of different computation models (continuous time, dataflow, discrete event). Detailed sensor extension. Supports TinyOS component design. *http://ptolemy.eecs.berkeley.edu*

† Includes support for parallel simulation.

*
C: Classical Models (e.g. TPC/IP, Ethernet)
A: Ad-hoc support (e.g. MANET protocols: AODV, DSR)
W: Wireless support (e.g. propagation, mobility, IEEE802.11)
WSN: Some common WSN protocols (e.g. Directed Diffusion, S-MAC)
$WSN+$: Rich WSN support with environment and battery models

TABLE II
GENERAL PURPOSE SIMULATION FRAMEWORKS

of WSN make many of the available models for classical tools useless.

B. *Specific WSN simulation frameworks*

Specific simulators target an overall system test, including the hardware and the operating system (OS). The main goal is to achieve a high level of fidelity. This *completeness* is their best advantage. Moreover, they have set an already clear trend in WSN simulation: the use of native sensor source code. It allows direct implementation and study of actual algorithms.

Currently, two different approaches show how to obtain *completeness*: TOSSIM and ATEMU (see Table III). TOSSIM replaces low-level components (radio, system clock, etc.) of Berkeley motes (sensors) by simulated models. The rest of the TinyOS components are cross-compiled from *native* code to simulated components. Thus, pure TinyOS applications execute on high fidelity simulated environments. However, such fidelity reduces scalability. Since it simulates at bit level, performance degrades as network load increases. ATEMU emulates mote AVR microprocessor instruction by instruction, while it simulates the radio model. Emulation of a complete hardware platform has two advantages: (1) the capability of testing OS and applications other than TinyOS and (2) the capability of simulating heterogeneous networks with different types of sensors. The penalties are the high processing requirements and the poor scalability.

	Platform	OS	Scalability	Heterogeneity	Additional notes
ATEMU	AVR processor	Any on an AVR processor	500 nodes	Yes, different sensors and applications	Supports monitoring of sensor node instruction by instruction. *http://www.cshcn.umd.edu/research/atemu/*
EmStar EmTOS	Microservers Berkeley Motes	Linux (FUSD) TinyOS	500 nodes	Yes, Mica and Microservers with different applications	Allows for combined use of simulated and real nodes. *http://cvs.cens.ucla.edu/emstar*
SNAP	SNAP processor	–	100,000 nodes (expected)	Yes, different code on SNAP processors	An asynchronous low power microprocessor designed for WSN with simulation capabilities. *http://vlsi.cornell.edu/sensor.php*
TOSSIM	Berkeley Motes	TinyOS	1000 nodes	No, only TinyOS/Mica running the same application	Widely used. In constant development. Bit-level granularity. *http://cs.berkeley.edu/∼pal/research/tossim.html*

TABLE III
SPECIFIC SIMULATION TOOLS

In spite of being a design goal, the achieved scale of the simulated networks with these tools (1000 nodes) is far from the expected size of a sensor net. SNAP (Table III) is a totally different approach intended to solve this situation. It is a microprocessor that can be used as the core of a deployed sensor, or to form an array of processors for parallel sensor simulation (up to 100,000 nodes). This way SNAP becomes a hardware, simulation and deployment platform.

IX. Conclusions

Despite the continuous effort made to support high fidelity models into WSN simulators, it is useless if scalability of such models is not faced. In this work, we identify the main scalability issues of WSN: radio channel and the integration of continuous-domain models. Both of them have an important impact on the performance of the simulated system, as demonstrated in Table I. Moreover, several experiments warn about sophisticated effects not yet being included into synthetic models. Including such effects may rise the scalability problem of WSN. More research is needed to bring new approaches to alleviate this problem.

In the authors opinion, the most promising solution is to combine lightweight mathematical abstractions of key parts of simulators, e.g., connectivity graphs instead of explicit computation of radio propagation, while only the protocols, or components under study should be implemented. Albeit, it requires a deeper characterization of real sensor networks components, not yet tackled.

X. Acknowledgments

This work has been funded by the Economy, Industry and Innovation Council, with the SOLIDMOVIL project (2I04SU044), by Fundacion Seneca with the ARENA Project (00546/PI/04), both of them from the Region of Murcia, and by the Spanish Research Council with the ARPaq project (TEC2004-05622-C04-02/TCM).

Three anonymous reviewers made relevant comments that help us improve this paper.

References

[1] R. Szewczyk, J. Polastre, A. Mainwaring, D. Culler, "Lessons from a sensor network expedition", in *Proc. First European Workshop on Sensor Networks (EWSN 2004)*, Berlin, Germany, January 2004.

[2] E. Egea-Lopez, J. Vales-Alonso, A. S. Martinez-Sala, P. Pavon-Mariño, J. Garcia-Haro, "Simulation tools for wireless sensor networks", in *Proc. Int. Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2005)*, Philadelphia, PA, pp. 559–566, July 2005.

[3] K. Pawlikowski, H. D. Joshua Jeong, J. S. Ruth Lee, "On credibility of simulation studies of telecommunication networks", *IEEE Communications Magazine*, vol. 40, no. 1, pp. 132–139, January 2002.

[4] D. Kotz, C. Newport, B. Gray, J. Liu, Y. Yuan, C. Elliot, "Experimental evaluation of wireless simulation assumptions", in *Proc. 7th ACM/IEEE Int. Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2004)*, Venice, Italy, pp. 78–82, October 2004.

[5] G. Zhou, T. He, S. Krishnamurthy, J. A. Stankovic, "Impact of radio irregularity on wireless sensor networks", in *Proc. 2nd Int. Conf. Mobile Systems, Applications and Services (MobySys)*, Boston, MA, pp. 125–138, June 2004.

[6] G. Riley, M. Ammar, "Simulating large networks: how big is big enough", in *Proc. First Int. Conf. on Grand Challenges for Modeling and Simulation*, San Antonio, TX, 2002.

[7] L. Zhong, J. Rabaey, A. Wolisz, "An integrated data-link energy model for wireless sensor networks", in *Proc. IEEE Int. Conf. on Communications*, Paris, France, 2004.

[8] A. Martinez-Sala, J. Molina-Garcia-Pardo, E. Egea-Lopez, J. Vales-Alonso, L. Juan-Llacer, J. Garcia-Haro, "An accurate radio channel model for wireless sensor networks simulation", Journal of Communications and Networks, vol. 7, no. 4, December 2005

[9] V. Naoumov, T. Gross, "Simulation of large ad hoc networks", in *Proc. ACM Modeling, Analysis and Simulation of Wireles and Mobile Systems (MSWiM 2003)*, San Diego, CA, pp. 50–57, 2003.

[10] E. Page, "Beyond speedup: PADS, the HLA and web-based simulation", in *Proc. Thirteenth Workshop on Parallel and Distributed Simulation*, Atlanta, GA, pp. 2–9, 1999.

[11] A. Cerpa, J. L. Wong, L. Kuang, M. Potkonjak, D. Estrin, "Statistical model of lossy links in wireless sensor networks", in *Proc. IEEE/ACM 4th Int. Conf. on Information Processing in Sensor Networks*, Los Angeles, CA, April 2005.

[12] A. Woo, T. Tong, D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks", in *Proc. ACM 1st Int. Conf. on Embedded Networked Sensor Systems*, Los Angeles, CA, pp. 14–27, November 200.

[13] A. Sridharan, M. Zuniga, B. Krishnamachari, "Integrating environment simulators with network simulators", *University of Southern California*, Dpt. of Computer Science Technical Report 04-386, 2004.

[14] M. Handy, D. Timmermann, "Simulation of mobile wireless networks with accurate modelling of non-linear battery effects", in *Proc. Int. Conf. on Applied Simulation and Modelling*, Marbella, Spain, pp. 532–537, September 2003.