Wireless sensor networks: modeling and simulation

Sajjad A. Madani
COMSATS Institute of Information Technology, Abbottabad
Pakistan

Jawad Kazmi
COMSATS Institute of Information Technology, Abbottabad
Pakistan

Stefan Mahlknecht
Vienna University of Technology, Vienna
Austria

Abstract

Although different modeling techniques have been proposed during the last 300 years, the differential equation formalism proposed by Newton and Leibniz has been the tool of choice for modeling and problem solving Taylor (1996); Wainer (2009). Differential equations provide a formal mathematical method (sometimes also called an analytical method) for studying the entity of interest.

Computational methods based on differential equations could not be easily applied in studying human-made dynamic systems (e.g., traffic controllers, robotic arms, automated factories, production plants, computer networks, VLSI circuits). These systems are usually referred to as discrete event systems because their states do not change continuously but, rather, because of the occurrence of events. This makes them asynchronous, inherently concurrent, and highly nonlinear, rendering their modeling and simulation different from that used in traditional approaches. In order to improve the model definition for this class of systems, a number of techniques were introduced, including Petri Nets, Finite State Machines, min-max algebra, Timed Automata, etc. Banks & Nicol. (2005); Cassandras (1993); Cellier & Kofman. (2006); Fishwick (1995); Law & Kelton (2000); Toffoli & Margolus. (1987).

Wireless Sensor Network (WSN) is a discrete event system which consists of a network of sensor nodes equipped with sensing, computing, power, and communication modules to monitor certain phenomenon such as environmental data or object tracking Zhao & Guibas (2004). Emerging applications of wireless sensor networks are comprised of asset and warehouse
management, automotive, home and building automation, civil infrastructure monitoring, healthcare, industrial process control, military battlefield awareness, and security and surveillance Cerpa et al. (2001).

As discussed earlier, modeling and simulation is a mean to verify the working and to measure the effectiveness of the different techniques proposed for WSNs. Analytical modeling provides quick insight for the techniques developed for WSNs but fail to give realistic results because of WSN specific constraints like limited energy and sheer number of sensor nodes Chen et al. (2006). Real world implementation and test beds are the most accurate method to verify the concepts but are restricted by costs, effort, and time factors as well as repeating environmental conditions is also not possible Zeigler (1976). Simulations provide a good approximation to verify different schemes and applications developed for WSNs at low cost and in less time. To have credible results through simulation, the choice of models and the simulation environment is important.

There is always a tradeoff between credible simulation results and the time required to get these simulation results. The results always depend upon the level of abstraction of the models. The more detailed is the model, the better the accuracy of results but higher the amount of time required for simulation.

The models used for simulation can have a significant impact on the overall simulation study. In this chapter, we will present a brief overview of the models available for different modules of sensor network simulation study in addition to the general-purpose simulation frameworks and tools that can be used to study WSNs. Such tools include NS-2, OMNeT ++, SenSim, NesCT, GlomoSim, OPNET Modeler, SENSE, Ptolemy II, VisualSense and J-Sim. Additionally, some WSNs specific simulators frameworks/emulators are also covered including TOSSIM, EmStar, ATEMU and PAWiS.

1. Introduction

Recent advances in MEMS and distributed computing has enabled WSNs powered diverse applications ranging from military to kindergartens. WSN is a network of sensor nodes equipped with sensing, computing, power, and communication modules to monitor certain phenomenon such as environmental data or object tracking Raghavendra et al. (2004). In current scenarios, the number of sensor nodes may be 20 to 30 but in future it may consist of $n^{th}$ power more sensor and actuator systems Dietrich et al. (2001). The position of the sensor nodes may not be pre-determined and require sensor nodes to be equipped with self organizing protocols Akyildiz et al. (2002). Generally, sensor nodes observe and sense the phenomenon with a sensing module, process the data with a computing module, and send the data to a required destination via wireless link with a communication module.

2. Wireless Sensor Network Applications

Emerging applications of wireless sensor networks are comprised of asset and warehouse management, automotives, home and building automation, civil infrastructure monitoring, healthcare, industrial process control, military battlefield awareness, and security and surveillance Zhao & Guibas (2004). Some examples of these application like robotic navigation Fu et al. (2009), aircraft corrosion monitoring Demo et al. (2010), direct load control in residential areas Molina-Garcia et al. (2007), personal mobile physiological monitoring and management system for chronic disease Toh et al. (2008), Wildfire detection Antoine-Santoni et al. (2006),
in-service motor monitoring and energy management Hu (2008), application in petrochemical industry Ke et al. (2008), application for critical infrastructure risk analysis of fossil fuelled power stations Isreb (2006) and tactical military applications Lee et al. (2009) can be found in the literature.

3. Historical Perspective

History of wireless communication used in the field can be traced back to late 1890’s to Anglo-Boer war in Namibia, the then German Southwest Africa, and was declared by IEEE as electrical engineering milestone in which Marconi’s system was used for wireless communication Sarkar et al. (2006). In 1921, the US Department of Defense initiated a radiotelegraphic network, which resulted in 125 stations network by 1925 Callaway et al. (2002). The Aloha system Abramson (1970) developed at the University of Hawaii is considered the first successful data network, which connected different campuses of the University of Hawaii. The Packet Radio Network Jubin & Tornow (1987), comprising of 138 network devices was developed in 1972. In 1997, IEEE 802 LAN/MAN Standard Committee released the first Wireless LAN standard. The development of Wireless Personal Area Networks began in 1997, with the formation of Home RF Working Group and with the formation of Bluetooth Special Interest Group in 1998 Callaway et al. (2002). Like other wireless networks, the sensor networks also has a long history with can be traced back to 1978 DARPA -sponsored Distributed Sensor Nets Workshop. Wireless Integrated Network Sensors (WINS) project was initiated in 1993 by University of California at Los Angeles. The University of California at Berkeley started PicoRadio project in 1999. AMPS project started in MIT, to develop a complete architecture for low power wireless sensor networks. LonWorks Dietrich et al. (2001) in 2001 and EIB: Installation Bus System Sauter et al. (2002) in 2002 was a big step towards building automation. Wireless Self Sustaining Sensor Network Project (WSSN) at institute of computer technology, Vienna University of technology aimed to establish sensor networks for building automation which are low power and self sufficient in energy.

4. Discre Discret Event Modeling and Simulation Methodologies

Modeling techniques for discrete event driven systems (including WSNs) are relatively recent. In this section, we present a non-comprehensive list of some of the formal modeling techniques created for modeling these systems.

4.1 Automaton

An automaton is defined as a graph representing system states and the transitions between them. The automaton receives a string of symbols as input, and it recognizes/rejects the inputs by advancing through the transitions. The input is read one symbol at a time; depending on the ending state, the automaton will accept or reject the input Cassandras (1993).

4.2 Timed automata

Timed automata, in particular, use clocks to describe the model’s timing behavior Alur & Dill (1994). The automaton is defined as a graph of states associated with clocks that determine

---

1. www.janet.ucla.edu
2. bwrc.eecs.berkeley.edu
3. www-mlt.mit.edu
the passage of time since the occurrence of an event. Every link is associated with a timing constraint that will define when the transition can be triggered. Whenever a transition executes, the associated clocks are reset. Timing constraints can also be associated with the model states, defining the duration of each of the states.

4.3 Finite state machines
Finite state machines (FSMs) can be represented as a graph in which the system’s behavior is defined as a finite set of nodes (the model’s states) and links between them (transitions between states). A given state reflects the evolution of the model, and transitions are associated with a given logical condition to enable the execution of the transition. When entering a state, an entry action can be executed (and an exit action can be executed when leaving it). Likewise, an input action can be triggered based on the current state and an input Cassandras (1993). An FSM is formally defined as

\[ FSM = (S, X, Y, f, g) \]  

Where
\( X \) = finite input set
\( Y \) = finite output set
\( S \) = finite state set

4.4 Markov chain
A Markov chain is a discrete-time stochastic model described using a graph. Models’ states are defined as nodes in the graph, and transitions between states are represented by links. One important property of Markov chains is that they are memoryless; thus, no state has a cause–effect relationship with the previous state. Therefore, knowledge of previous states is irrelevant for predicting the probability of the future states.

4.5 Generalized Semi-Markovian Process
A Generalized Semi-Markovian Process (GSMP) is a stochastic process (i.e., a collection of random variables over a probability space indexed by time). A GSMP is based on the notion of a state that makes a transition when an event associated with the current state occurs, and the state space is generated by a stochastic timed automaton Glynn (1989). Several possible events can compete to trigger the next transition, and each of these events has its own probabilistic distribution for determining the next state.

4.6 Petri nets
Petri nets Peterson (1981) define the structure of concurrent systems using a bipartite graph. One type of the graph’s nodes, the places, represents the system states, and the second kind, the transitions, represents the net evolution Adi Mallikarjuna et al. (2007).

4.7 Queuing networks
Queuing networks are based on a customer–server paradigm, in which customers make service requests to the servers and these requests are queued at the server until they can be serviced. The arrival time for customers and the service time at a server are described as stochastic models. By defining the number of servers and the buffering capacity on each of them, we can determine performance metrics (including the number of customers in line,
throughput—number of customers serviced per time unit, turnaround times, etc.). Different policies can be used (priorities; preemption; first in, first out; etc.) Li & Li (2003).

4.8 Calculus of communicating systems
The formal language of calculus of communicating systems (CCS) provides primitives for concurrency and parallelism, based on synchronous communications between exactly two components. The language expressions are interpreted as a labeled transition system, and bi-simulation can be used to prove equivalence of models Hansson & Jonsson (1990).

4.9 Temporal logic
Temporal logic is a system of rules and symbols used for representing propositions that can include the timing properties of the system Manna & Pnueli (1992). It consists of a logic set of propositions that view time as a sequence of states and that can be true or false according to their state and their time of occurrence. Temporal logic has been used to verify formally timed automata. The idea is to check predictability of certain conditions according to the time that they occur, conditions that might eventually arise, or others that are guaranteed not to occur.

4.10 Communicating sequential processes
Communicating sequential processes (CSP) is a formal language based on process algebra that has been widely used to model concurrent systems Hoare (1985). Models are described using independent processes that interact with each other through message-passing representing the occurrence of events Wainer (2009).

4.11 Specification and Description Language
Specification and Description Language (SDL) was created to specify in a nonambiguous way the behavior of real-time applications. It was originally focused on communication systems, by providing a graphical and textual representation with equivalent semantics. A system is defined as a set of extended FSMs that can be interconnected Belina & Hogrefe (1989).

4.12 Event graphs
Event graphs are oriented graphs that represent the organization of the events of a discrete event system Schruben et al. (2003). Events constitute nodes of the graph; that is, the vertices represent the state transition functions, and the links between nodes capture the scheduling of such events. Each link starts at the node performing the scheduling operation (which represents an event), and it ends at the node representing the event to be scheduled. Each scheduling relationship has an associated delay and condition (a Boolean function of the state), and an event is scheduled only when the condition is true Wainer (2009).

4.13 Systems theory
Systems-theoretical approaches derive from systems theory von Bertalanffy (1969). Systems theory represents every entity under study using the concept of system, which is seen as a collection of objects and their interactions. In systems theory, the system’s global behavior is seen as a composition of the individual behavior of the components, and we can find emergent behavior that is not explicitly defined in the parts of the system. Systems theory is based on the idea that every phenomenon can be viewed as a mathematical relationship among a set of entities in the system. The theory is generic and tries to find common behavior and properties
in different fields of study (for instance, hydraulics, economy, biology, or social sciences), thus providing a unified view of science and engineering.

### 4.14 Discrete-Event Systems Specifications

Discrete-Event Systems Specifications (DESS) formalism Zeigler (1976; 1990) is a mathematical modeling technique derived from systems theory that allows the construction of hierarchical and modular models, providing a well-defined coupling of components. Given its hierarchical nature, DEVS allows the coupling of existing models modularly, allowing us to build complex systems from simple ones. DEVS theory provides a rigorous methodology for representing models, and it presents an abstract way of thinking about the world independently of the simulation mechanisms Wainer (2009).

### 5. Phases in a Simulation Study

There have been different kinds of life cycles proposed for studies in modeling and simulation. In this section, we summarize the basic steps that should be considered in doing a simulation study. The life cycle does not have to be interpreted as strictly sequential; it is iterative by nature, and sometimes transitions in opposite directions can appear. Likewise, some of the steps can be skipped, according to the complexity of the application. It is highly recommended to use a spiral cycle with incremental development for steps 2–8 (Section 5.2 through Section 5.8), which can cause a revision to earlier phases. Each phase in the spiral cycle should end with a working prototype including more functionality than the previous cycle:

#### 5.1 Problem formulation

The simulation process begins with a practical problem that requires solving or understanding. It might be the case of a cargo company trying to develop a new strategy for truck dispatching or an astronomer trying to understand how a nebula is formed. At this stage, we must understand the behavior of the system of interest (which can be a natural or artificial system, existing or not), organizing the system’s operation as objects and activities within the experimental framework of interest. Then we need to analyze different alternatives of solutions by investigating other previously existing results for similar problems. The most acceptable solution should be chosen (omitting this stage could cause the selection of an expensive or wrong solution). We also must identify the input/output variables and classify them into decision variables (controllable) or parameters (non-controllable). If the problem involves performance analysis, this is the point at which we can also define performance metrics (based on the output variables) and an objective function (i.e., a combination of some of the metrics). At this stage, we can also do risk analysis and decide whether to follow or discard the project.

#### 5.2 The Conceptual Model

This step consists of building a high-level description of the structure and behavior of the system and identifying all the objects with their attributes and interfaces. We also must define what the state variables are, how they are related, and which ones are important for the study. In this step, key aspects of the requirements are expressed (if possible, using a formalism, which introduces a higher degree of precision). During the definition of the conceptual model, we need to reveal features that are of critical significance (e.g., possibility of instability, deadlock, or starvation). We must also document nonfunctional information—e.g., possible future changes, nonintuitive (or non-formal) behavior, and the relation with the environment.
5.3 The collection and analysis of input/output data phase
In this phase, we must study the system to obtain input/output data. To do so, we must observe and collect the attributes chosen in the previous phase. When the system entities are studied, we try to associate them with a timing value. Another important issue during this phase is the selection of a sample size that is statistically valid and a data format that can be processed with a computer. Finally, we must decide which attributes are stochastic and which are deterministic. In some cases, there are no data sources to collect (for instance, for non-existing systems). In those cases, we need to try to obtain data sets from similar systems (if available). Another option is to use a stochastic approach to provide the data needed through random number generation.

5.4 Modeling phase
In the modeling phase, we must build a detailed representation of the system based on the conceptual model and the I/O data collected. The model is built by defining objects, attributes, and methods using a chosen paradigm. At this point, a specification model is created, including the set of equations defining its behavior and structure. After finishing this definition, we must try to build a preliminary structure of the model (possibly relating the system variables and performance metrics), carefully describing any assumptions and simplifications and collecting them into the model’s EF.

5.5 Simulation phase
During the simulation stage, we must choose a mechanism to implement the model (in most cases using a computer and adequate programming languages and tools), and a simulation model is constructed. During this step, it might be necessary to define simulation algorithms and to translate them into a computer program. In this phase, we also must build a model of the EF for the simulation.

5.6 Verification and validation
During the previous steps, three different models are built: the conceptual model (specification), the system’s model (design), and the simulation model (executable program). We need to verify and validate these models. Verification is related to the internal consistency among the three models (is the model correctly implemented?). Validation is focused on the correspondence between model and reality: are the simulation results consistent with the system being analyzed? Did we build the right model? Based on the results obtained during this phase, the model and its implementation might need refinement. As we will discuss in the next section, the V&V process does not constitute a particular phase of the life cycle, but it is an integral part of it. This process must be formal and must be documented correctly because later versions of the model will require another round of V&V, which is, in fact, one of the most expensive phases in the cycle.

5.7 Experimentation
We must execute the simulation model, following the goals stated in the conceptual model. During this phase, we must evaluate the outputs of the simulator, using statistical correlation to determine a precision level for the performance metrics. This phase starts with the design of the experiments, using different techniques. Some of these techniques include sensitivity analysis, optimization, variance reduction (to optimize the results from a statistical point of view), and ranking and selection (comparison with alternative systems).
5.8 Output analysis phase
In the output analysis phase, the simulation outputs are analyzed in order to understand the system behavior. These outputs are used to obtain responses about the behavior of the original system. At this stage, visualization tools can be used to help with the process. The goal of visualization is to provide a deeper understanding of the real systems being investigated and to help in exploring the large set of numerical data produced by the simulation.

6. M&S Tools and Environments for WSNs
Different schemes developed for Wireless Sensor Networks (WSNs) are verified by analytical techniques, simulations, and test beds. Cinque et al. (2007); Lopez et al. (2005). Analytical modeling may provide quick insight but fail to give realistic results for reasons like limited energy, memory, and processing power, and sheer number, unattended operation, harsh environments of sensor nodes Chen et al. (2006). Real world implementation and test beds are the most accurate ways to study WSNs but such methods requires huge effort, time and money. Simulations provide a good approximation to verify different schemes and applications developed for WSNs at low cost and in less time. To have credible results through simulations, correct modeling and the selection of simulation tool plays a vital role. In the following sections, we present brief description of different models as well as simulation tools available to study WSNs.

6.1 Simulation Tools and Environments

6.1.1 NS-2
NS-2\(^5\) is a discrete event, object oriented, general purpose network simulator written in C++. It is the most widely used simulator Kurkowski et al. (2005). Its main focus is IP networks. To simulate WSNs with more or less 100 nodes, NS-2 can be a good choice because of its large community but for 100+ nodes, it is no more scalable Naoumov & Gross (2003). “One of the problems of ns2 is its object-oriented design that introduces much unnecessary inter-dependence between modules. Such interdependence sometimes makes the addition of new protocol models extremely difficult, which can only be mastered by those who have intimate familiarity with the simulator.” Chen et al. (2006). SensorSim Park et al. (2000) is a NS-2 based simulator for modeling sensor networks. Some WSN specific features are included but because of the “unfinished nature of the software”\(^6\), the simulator is no longer available.

6.1.2 OMNeT++
OMNeT ++ Varga (2001) is a discrete event, component based, general purpose, public source, modular simulation framework written in C++. It provides a strong GUI support for animation and debugging. Mobility framework (MF) Drytkiewicz et al. (2003) for OMNeT++ is specific purpose add-on to simulate ad-hoc networks. The lack of a WSN specific module library Lopez et al. (2006) may be a problem currently but many research groups are working to add WSN specific additional modules. SenSim Mallanda et al. (2005) is OMNeT based simulation framework for WSN. It provides the basic implementation of different hardware (e.g., basic radio, and CPU) and software (simple routing schemes) modules for WSN. It provides a template with basic implementation or empty body which can help anyone to jump start simulating WSNs Egea-Lopez et al. (2006).

\(^5\) www.isi.edu/nsnam/ns
\(^6\) nesl.ee.ucla.edu/projects/sensorsim
6.1.3 NesCT
NesCT\(^7\) is an add-on for OMNeT++ which allows simulation of TinyOS based sensor networks in OMNeT (language translator between OMNeT and TinyOS implementations).

6.1.4 PAWIS
PAWIS\(^8\) simulation framework is OMNeT plus plus based discrete event simulation framework. It provides a rich library of modules and supports mobility and environmental dynamics. It also provides a simulation template for users to quick start simulation study Madani et al. (2008); Weber et al. (2007).

6.1.5 GlomoSim
Global Mobile Information System Simulator (GlomoSim) Zeng et al. (1998) is a library based general purpose, parallel simulator written in Parsec\(^9\). It can simulate up to 10,000 nodes Lopez et al. (2005) and can be very useful in studying large scale WSNs. GlomoSim is superseded by QualNet\(^10\), a commercial network simulator. sQualNet Varshney et al. (2007), an evaluation framework for sensor networks, based on QualNet is released recently.

6.1.6 OPNET
OPNET Modeler\(^11\) is general purpose, object oriented, C-based discrete event simulation environment. It’s commercial and therefore not used widely. It comes with a version for academic use but with limited capabilities. OPNET is a large and powerful software with a wide variety of possibilities. OPNET can be used as a research tool and also as a network design/analysis tool. OPNET was originally built for the simulation of fixed networks, and therefore, it contains extensive libraries of accurate models from commercially available fixed network hardware and protocols Cai & Jia (2009); Hammoodi et al. (2009); Hasan et al. (2009); Zhuo et al. (2007).

6.1.7 SENSE
SENSE Chen et al. (2006) is a sensor network specific, component based simulator written in C++ built on the top of COST Chen & Szymanski (2002). Parallel simulation can be done to study large scale WSNs. It provides basic implementations like AODV and DSR as well as oversimplified models for batter and energy consumption.

6.1.8 Ptolemy II and J-Sim
Ptolemy II Liu et al. (2001) and J-Sim Miller et al. (1997); Sobeih et al. (2005; 2006) are general purpose, component based simulation frameworks written Java. Both simulation frameworks provide a rich support for WSNs.

6.1.9 Cell-DEVS
The Cell-DEVS formalism allows defining cellular models based on the discrete-event system specification. Cell-DEVS allows defining asynchronous cell spaces with explicit timing definition. This approach is still based on the formal specifications of DEVS, but it allows the user to

---

\(^7\) www.omnetpp.org
\(^8\) www.ict.tuwien.ac.at/pawis
\(^9\) pcl.cs.ucla.edu/projects/parsec
\(^10\) www.scalable-networks.com
\(^11\) www.opnet.com
focus on the problem to be solved by using simple rules for modeling (like with CA). Explicit
timing delay constructions can be used to define precise timing in each cell. This approach
allows enhancing the modeling experience in different aspects. In terms of performance, only
active cells execute their local computing function, and the execution results are spread out
after a predefined delay (only if a state change has occurred). The delay function provides a
natural mechanism for defining timing information.
The modeling technique permits keeping the ability of CA to describe complex systems using
very simple rules, while also permitting us to bridge the gap between a discrete time
and a discrete event description like DEVS. The use of DEVS as the basic formal specification
mechanism enables us to define interactions with models defined in other formalisms.
Individual cells can provide data to those models; integration between them could enable
defining of complex hybrid systems and multimodels developed with different techniques
and integrated through a DEVS interface. This approach provides "evolvability" of the models
through a technique that is easy to understand and to map into other existing techniques,
while having the potential of evolving into complex models Qela et al. (2009).

6.1.10 GTNetS
The "Georgia Tech Network Simulator," (GTNetS) 12 developed and maintained by the De-
partment of Electrical and Computer Engineering at Georgia Tech is an object-oriented design
written completely in C++. The design of GTNetS matches closely the design of actual net-
work protocol stacks and other network elements. Further, GTNetS was designed from the
beginning to run a distributed environment, leading to better scalability Cheng et al. (2006);

6.1.11 SystemC
SystemC is a C++ based modeling platform supporting design abstractions at the register-
transfer, behavioral, and system levels. The SystemC classes add the necessary constructs to
C++ for modeling systems and hardware at various levels of abstraction-from the abstract
untimed models to cycle-accurate RTL models. The power of SystemC is that it can be used as
a common language by system designers, software engineers, and hardware designers Rafiee
et al. (2009); Vasilevski et al. (2007).

6.1.12 Prowler
Prowler 13 is an event-driven wireless network simulator designed to run in Matlab envi-
ronment. The simulator, written originally to simulate Berkeley MICA motes, is extendable
also for more general platforms. Prowler is implemented in Matlab language (m-file) which
makes direct simulation code, e.g., routing protocol or application, interchange between sim-
ulator and sensor platforms impossible. Benefits gained from Matlab environment are easy
prototyping of applications, integration of different optimization algorithms, GUI interface
and good visualization capabilities.

6.1.13 NCTUns2.0
NCTUns2.0 14 is a discrete event simulator whose engine is embedded in the kernel of a UNIX
machine. The actual network layer packets are tunneled through virtual interfaces that simu-

---

12 http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/
13 http://www.isis.vanderbilt.edu/projects/nest/prowler
14 http://nsl.csie.nctu.edu.tw
late lower layers and physical devices. This notable feature allows simulations to be fed with real program data sources. A useful GUI is available in addition to a high number of protocols and network devices, including wireless LAN. Unfortunately, no specific designs for WSN are included.

6.1.14 JiST/SWANS
JiST/SWANS is a discrete event simulation framework that embeds the simulation engine in the Java bytecode. Models are implemented in Java and compiled. Then, bytecodes are rewritten to introduce simulation semantics. Afterwards, they are executed on a standard JVM. This implementation allows the use of unmodified existing Java software in the simulation, as occurs with NCTUns2.0 and UNIX programs.

6.1.15 SSFNet
SSFNet is a set of Java network models built over the Scalable Simulation Framework (SSF). SSF is a specification of a common API for simulation, that assures portability between compliant simulators. There are multiple Java and C++ implementations of SSF. DartmouthSSF (DaSSF) [30], for instance, is a C++ implementation of SSF oriented to (parallel) simulation of very large scale communication networks.

6.1.16 Ptolemy II
Ptolemy II is a set of Java packages that support different models of simulation paradigms (e.g. continuous time, dataflow, discrete-event). It also addresses the modeling, simulation and design of concurrent, real-time and embedded systems.

6.2 Emulation Tools
In addition to the above cited simulation tools/environments for WSNs, there are number of emulation tools/environments as well. Some of the these are TOSSIM Levis et al. (2003), EmStar Girod et al. (2004), and ATEMU Polley et al. (2004). Such tools come with inherent advantage. Whatever code is used for simulation/emulation, the same code is used on the real sensor node with slight modifications. It also provides detailed information about resource utilization. The main problem with such frameworks is “the user is tied to a single platform either software or hardware (typically MICA motes), and to a single programming Language (typically TinyOS/NesC)” Lopez et al. (2005).

6.3 Other Tools
There are many tools which do not fall into either simulation or emulation catagories like TEPAWSN Man et al. (2009) which is a tool environment for Wireless Sensor Networks.

7. Summary
Wireless Sensor Networks is an emerging field with many application in almost all walk of life. Researchers are actively involved in the development of new and improving the existing techniques and technologies for making the life more easier. Each of these developed and/or improved techniques need to be extensively tested and varified before it can be used in the

http://jist.ece.cornell.edu/
http://www.ssfnet.org
http://ptolemy.berkeley.edu/ptolemyII/
actual production. Most accurate and reliable method is, of-course, the real-world implementation and test-beds. But these are sometimes not possible or even harder. In that case the method of the choice is the modeling and simulation.

In this chapter we presented a partial list of available models and simulation/emulation tools for the wireless sensor networks, available. The chapter started with an introduction and the historical perspective to the field of WSN and some of its applications. We then briefly discussed the different discrete event modeling and simulation methodologies after which the major steps/phases in a M&S study are outlines with the identification of major milestone. The chapter concludes with a list of different models, simulation and emulation tools.

8. References


http://ait.upct.es/~eegea/


http://citeseer.ist.psu.edu/naoumov03simulation.html

and Mobile Systems.
URL: http://portal.acm.org/citation.cfm?id=346870


