Discrete Event Modeling and Simulation of Wireless Sensor Network Performance

T. Antoine-Santoni
J.F. Santucci
E. De Gentili
B. Costa
University of Corsica
UMR CNRS 6134 Quartier Grossetti
BP 52, 20250 Corte, France
antoine-santoni@univ-corse.fr

The wireless distributed microsensor networks have profited from recent technological advances and it seems essential to precisely understand these systems. Modeling and simulation appear to be an essential aspect of predicting the Wireless Sensor Network (WSN) specific behavior under different conditions. We want to provide a new method of modeling, simulation and visualization of WSN using a discrete-event approach. Described by Zeigler in the 1970's, the Discrete Event System Specification is ideal for describing the asynchronous nature of the events occuring in WSN. We have provided a basic model for the analysis of WSN performance, including routing management, energy consumption and relative CPU activity. Our approach uses a detailed definition of node-oriented components and aims to introduce methods of visualizing the network at a different level of abstraction.

Keywords: modeling, simulation, visualization, DEVS, WSN, performance

1. Introduction

Advances in hardware technology and engineering design have led to reductions in size, power consumption and cost. This has led to the production of compact autonomous nodes, each containing one or more sensors, computation and communication capabilities and a power supply. Networks of wireless sensors are the result of rapid convergence of the following three key technologies [1].

- Computing/Internet: Computing power is becoming small and inexpensive enough to add to almost any object. Networks of computers facilitate collaboration through information and resoure sharing.
- Sensor: Miniaturization and micromachining leads to smaller sizes, low power and lower costs, allowing monitoring with higher granularity. There currently exist many types of sensors and more are being developed.
- Wireless/Antennas: Spans a host of technologies including Bluetooth and WiFi networks, cellular and satellite communications.

These recent technologial advances have led to the definition and use of Wireless Sensor Network (WSN). The sensor nodes are usually scattered in a sensor field [2] as shown in Figure 1. Each of these scattered sensor nodes is capable of collecting data and routing data back to the sink. Data are routed back to the sink by a multihop infrastructureless architecture through the sink. The sink may communicate with the task manager node via Internet or satellite. The design of the sensor network as depicted by Figure 1 is influenced by many factors including fault tolerance, scalability, production costs, operating environment, sensor network topology, hardware constraints, transmission media and power consumption.

A sensor node combines the abilities to compute, to communicate and to sense [3]. In a sensor network, different functionalities can be associated with the sensor nodes [4]. In earlier works, all sensor nodes are assumed to be homogeneous, having equal capacity in terms of com-

SIMULATION, Vol. 84, Issue 2/3, February/March 2008 103–122
© 2008 The Society for Modeling and Simulation International
DOI: 10.1177/0037549708091641
Figures 11, 15, 16 appear in color online: http://sim.sagepub.com

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016

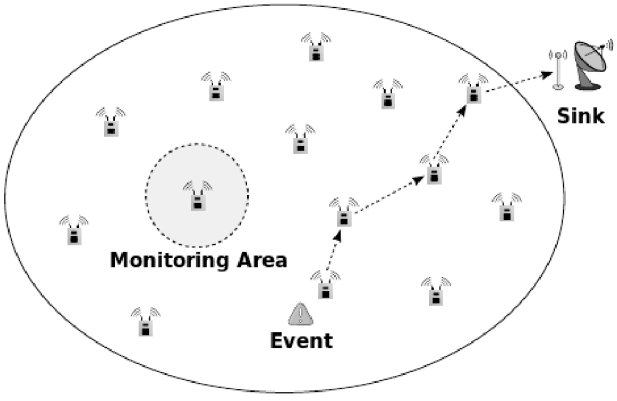


Figure 1. Sensor nodes in a sensor field

putation, communication and power. However, depending on the application, a node can be dedicated to a particular special function such as relaying or aggregation.

The goal of sensor is to send collected data, usually via radio transmitter, to a command center (sink or Base Station) either directly or through a data concentration center (a gateway). Based on the node descriptions in [5, 2], the main components of the sensor consist of a sensing unit, a processing unit, a transceiver and a power unit as depicted in Figure 2.

For the user to understand the behavior of a Wireless Sensor Network, the following five points should be known.

1. The communication tool is represented by the antenna and its role is to send some information to the channel.

2. Memory is the unit of storage of information evolving in the node. Information has two sources: from another node and from the sensor board in an environmental monitoring case. In both cases, information is treated by processor.

3. The processor treats all information in the node. The CPU manages activity in the mote and reacts according to instruction type.

4. The battery defines the lifetime of a node. Each component consumes energy according to its realized action. Energy is also consumed in sleep state.

5. The sensor board regroups monitoring activities. It can transmit information collected by the sensor but also transmits a message alert if a critical threshold is reached.

Modeling and simulation appear to be an essential part of understanding the behavior of a WSN under specific conditions. The network simulation for sensors is a challenging problem as it has to accurately model the hardware and energy constraints typical of sensor nodes and also the various aspects particular to sensor networks. The hierarchical nature of Discrete Event System Specification (DEVS) makes it perfect for describing a system such as sensor mote. The discrete-event nature improves the execution performance of a model like this due to the asynchronous nature of the events occurring in WSN.

Some research on modeling wireless ad hoc networks using DEVS has been carried out previously. In [6], the authors describe how to use the Cell-DEVS formalism in order to model routing protocol Ad-hoc On-Demand Distance Vector (AODV). DEVS is used to formally specify discrete-events systems using a modular description [6]. This strategy allows the re-use of tested models, improving the safety of the simulations and allowing development time to be reduced. As it is discrete-event formalism, it uses a continuous time base which allows accurate timing representation and reduces CPU time requirements. This very interesting work leans on the DEVS formalism in order to study the routing in wireless ad hoc networks.

A coupling between the NS-2 simulator and the DEVS formalism has been presented [7]. NS, also popularly called NS-2 in reference to its current generation, is a discrete-event network simulator [8]. The behavior of a sensor node's application and its environmental behaviors such as battlefields have been defined using DEVS modeling [7]. Furthemore, the authors point out the roles of networking protocol behaviors which are assigned to NS-2 since NS-2 has well-designed network protocol libraries. However, there are no modular aspects concerning the components involved in the sensor's behavior and thus its seems difficult to implement a specific environmental scenario. According to these previous remarks, we choose to define all components of Wireless Sensor Network using DEVS formalism.

The rest of the paper is organized as follows. Section 2 introduces briefly the Wireless Sensor Network area. In Section 3 we present the DEVS formalism. Section 4 presents the DEVS formalism-based approach in order to describe the behavior of Wireless Sensor nodes. The implementation and the validation of the proposed approach with results from simulation examples are detailed in Section 4. Finally, in Section 5 we present some conclusions and comment on future research.

2. Overview of WSN Simulation

Wireless Sensor Network research currently has many different foci and several fields of applications, such as channel access control, routing protocol definition, network management, QoS, energy consumption and CPU activity. There exist different simulators for representing activity and performance of WSN.

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016

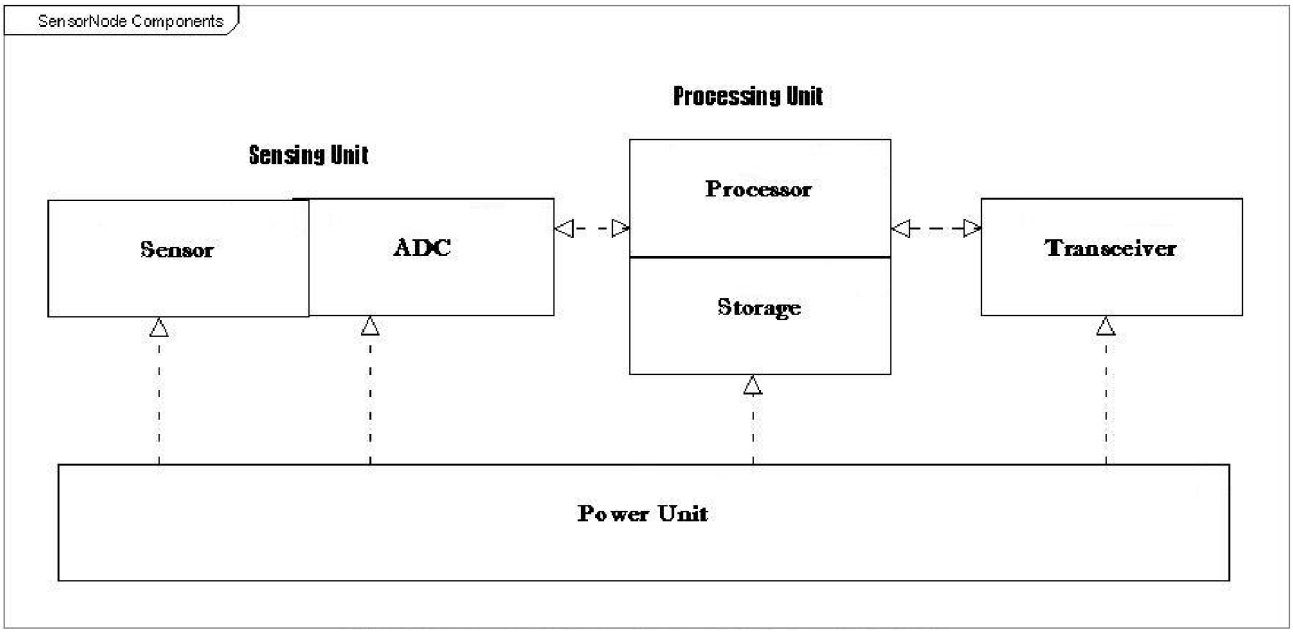


Figure 2. Components of a sensor node

SensorSim [9] extends the NS-2 network simulator with models of sensor channels and accurate battery and power consumption. Each node has a sensor stack that acts as a sink to the signals from each.

Atemu [10] is a software emulator for AVR processor-based systems. Along with support for the AVR processor, it also includes support for other peripheral devices on the MICA2 sensor node platform such as the radio. Atemu can be used to perform high-fidelity large-scale sensor network emulation studies in a controlled environment. Although the current release only includes support for MICA2 hardware, it can easily be extended to include other sensor node platforms. It allows for the use of heterogeneous sensor nodes in the same sensor network. Atemu cannot represent different activities of hardware components as it uses a high abstraction level.

TOSSIM [11] and PowerTOSSIM [12] are two important simulators which can correctly describe routing protocol, node applications or energy consumption but they are strongly dependent upon TinyOS and cannot represent generic framework of heterogeneous platforms.

Glonemo [13] can be considered similar to that presented here. Indeed, Glonemo introduces some solutions such as the MAC layer for describing a wireless sensor node. However, certain parameters appear uncertain such as CPU activity, general energy consumption and sensing activity.

SENS [14] is an application-oriented wireless sensor network which models ad hoc static nodes. It provides models for a limited set of sensors and actuators, a model for the environment and a framework for testing applications. SENS appears to be a suitable WSN simulator; however, some characteristics such as the addition of new sensor models or modeling arbitrary ubiquitous computing environments are missing.

It is particularly difficult to find a generic, easily customizable, modular simulator or a model able to represent the behavior of a node and generate particular environmental scenarios. We highlight the different aspects of each existing approach in Table 1.

We require a simulator able to represent a sensor node at different abstraction levels and be able to describe component behavior in particular conditions. Modular aspects of components in sensor models do not exist. In this paper, we focus on the representation capacity of DEVS formalism. The possibility of distinguishing different abstraction levels is clearly essential to allow the definition of the activity of the node components as well as general behavior in the network.

3. DEVS Formalism

Based on systems theory, DEVS formalism was introduced by Zeigler in the late 1970s [15]. It provides a hierarchical and modular method of modeling the discrete-event systems. A system (or model) is called modular if it possesses the input and output ports permitting interaction with its outside environment. In DEVS, a model is seen as a 'black box' S which receives and broadcasts messages

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016

Table 1. Comparison of existing approachs

	Components	Customizable	Modular	Generic	Environment	Topology	Energy
TOSSIM/Power TOSSIM	++	–	–	– –	–	– –	+
SensorSim	–	+	+	+	–	+	+
SENS	–	+	++	++	+	Fuzzy	–
Glonemo	+	–	–	–	++	–	– –
ATEMU	–	–	++	– –	–	–	– –
Avrora	+	–	–	–	–	–	++

\+ this characteristic is present in the tool
++ a great quality of the tool
– characteristic not really defined
– – a great deficiency of the tool
Fuzzy not clearly defined or explained

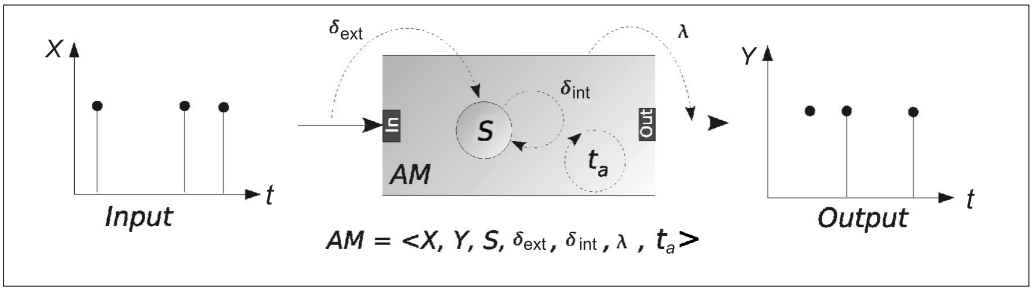


Figure 3. DEVS atomic model

on its input and output ports. This section deals with the basic notions of the DEVS formalism.

DEVS formalism [15] provides a means of specifying a mathematical object called a system. Basically, a system has a time base, inputs, states, outputs and functions for determining the next states and outputs given current states and inputs [15]. DEVS formalism is a simple way of characterizing how discrete-event simulation languages specify discrete-event system parameters. It is more than just a means of constructing simulation models. It provides a formal representation of discrete-event systems capable of mathematical manipulation in the same way as differential equations. Furthermore, by allowing an explicit separation between the modeling and simulation phases, the DEVS formalism is the best way to perform an efficient simulation of complex systems using a computer.

In DEVS formalism, one must specify (1) basic models from which larger models are built, and (2) how these models are connected in hierarchical fashion. Basic models (called atomic models) are defined by the structure

$$AM = \langle X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, t_{\text{a}} \rangle$$

where X is the set of input values, S is the set of sequential states, Y is the set of output values, δ_{int} is the internal transition function dictating state transitions due to internal events, δ_{ext} is the external transition function dictating state transitions due to external input events, λ is the output function generating external events at the output and t_{a} is the time-advance function associating a life time to a given state.

Figure 3 represents an atomic model AM with its output data Y calculated according to input data X. The atomic model AM has a state variable S that can be reached during the simulation. The functions δ_{ext}, λ, δ_{int} and t_{a} represent the model change of state when an external event occurs on one of those outputs (external transition function), the disposal of the output Y (output function), the model change of state after having given an output (internal transition function) and the determination of the duration of the model state (time-advance function), respectively.

We illustrate the behavior of an atomic model as follows. The external transition function describes how the system changes state in response to an input. When an input is applied to the system, it is said that an 'external even' has occurred. The new state s' is then calculated according to the current state s. The internal transition function describes the autonomous (or internal) behavior of the system. When the system changes state autonomously, an internal event is said to have occurred. The output function generates the outputs of the system when an internal transition occurs. The time-advance function determines the amount of time that must elapse before the next inter-

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016

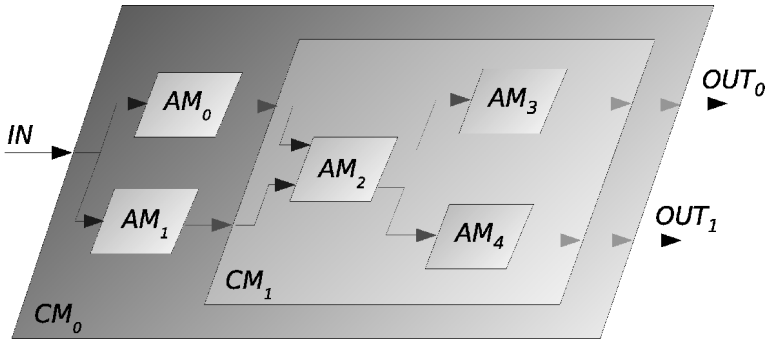


Figure 4. DEVS coupled model

nal event will occur, assuming that no input arrives in the interim.

An atomic model allows the behavior of a basic element of a given system to be specified. Connections between different atomic models can be performed by a coupled model (CM) [15]

$$CM = \langle X, Y, D, \{M_i\}, \{I_i\}, \{Z_{i,j}\} \rangle$$

where X is the set of input values, Y is the set of output values and D is the set of model references. For each $i \in D$, M_i is an atomic model, I_i is the set of influences of the model and $Z_{i,j}$ is the i to j translation function (output function).

A coupled model describes how to couple (connect) several component models to form a new model. This new model can itself be employed as a component in a larger coupled model, thus giving rise to hierarchical construction. The coupled models are defined by a set of sub-models (atomic and/or coupled) and express the internal structure of the system's sub-parts due to the coupling definition between the sub-models.

Figure 4 shows an example of the hierarchical structure of coupled model CM_0 which has an input port IN and two output ports OUT_0 and OUT_1. It contains the atomic sub-models AM_0, AM_1 and the coupled model CM_1. The latter can encapsulate other models such as atomic models AM_2, AM_3 and AM_4. A coupled model is specified through the list of its components (AM_0, AM_1, AM_2, AM_3, AM_4 and CM_1), the list of its internal couplings ($AM_0 \rightarrow CM_1$ and $AM_1 \rightarrow CM_1$), the list of the external input couplings ($IN \rightarrow AM_0$ and $IN \rightarrow AM_1$), the list of the external output couplings ($CM_1 \rightarrow OUT_0$ and $CM_1 \rightarrow OUT_1$) and the list of the sub-model's influence ($CM_1 = \{AM_0, AM_1\}$ or CM_1 influenced by AM_0 and AM_1).

DEVS formalism is mainly used for the description of discrete-event systems. It constitutes a powerful modeling and simulation tool yielding a system modeling on several levels of description as well as the definition of the behavior of the models. One of DEVS formalism's important properties is that it automatically provides a simulator for each model. DEVS establishes a distinction between a system modeling and a system simulation so that any model can be simulated without the need for a specific simulator to be implemented. Each atomic model is associated with a simulator in charge of managing the component's behavior and each coupled model is associated with a coordinator in charge of the time synchronization of underlying components. A simulator is associated with the DEVS formalism in order to exercise the coupled model's instructions to actually generate its behavior. The architecture of a DEVS simulation system is derived from the abstract simulator concepts associated with the hierarchical and modular DEVS formalism.

One of the main interests in DEVS formalism is the fact that it allows an explicit separation between the modeling and simulation part. This means that we can define the model representing the behavior of a given system without having to consider the simulation phase.

4. WSN Modeling Approach

It seems important to represent the different basic hardware components of the node. We have developed a generic approach allowing the definition of out-of-context behaviors of components in order to re-use this behavior in a context-in manner [16]. A context-out model is an abstraction of a model. It represents a behavior allowing it to be stored in a model library. A context-in model is a context-out model extracted from a library and formatted in order to be directly re-usable in its environment.

This generic approach leads us to define the behavior of different components. We try to delimit the different reactions of the node units to move towards the description of a general behavior of a sensor using a DEVS formalism. The advantages of this formalism for describing complex system are evident in the research; however, a definition of a sensor network and in particular sensor node do not exist.

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016

As sensor networks gain more importance in the research communities, it is crucial to show the advantages of DEVS formalism and to have a simulator with a modular structure. The use of this formalism in accordance with its definition implies, for this research area, two essential points: (1) a modeling specification step and consequently a clear interpretation of simulations results in the real world and (2) a non-ambiguous operational semantic step allowing the introduction of a formal specification of mechanics of simulation using an abstract simulator.

In this section we will first deal with the kinds of messages which are going to be exchanged between models of a WSN in Section 4.1. We will then introduce the different atomic and coupled models required in order to model the behavior of a WSN. These models are derived from the components of a given node of the WSN as shown in Figure 2. Section 4.2 is dedicated to the description of an atomic model called AM COM which has been defined in order to represent the communication involved in a node of the WSN. The coupled model, called CM Processor, is a key component in dealing with routing information and is explained in Section 4.3. We then describe the atomic models: AM Battery in Section 4.4; AM Memory in Section 4.5; and AM Sensorboard and AM Env in Section 4.6. These allow us to deal with the behavior of the battery, the memory and the environmental interaction involved in a sensor. Finally, the description of coupled model SENSOR which represents the overall behavior of a sensor due to the couplings involved can be found in Section 4.7.

4.1 Messages involved in WSN Modeling

The messages which are exchanged between the components of a WSN involve different kinds of information in order for the simulation to be able to efficiently describe the behavior of such a network. A message involves the nine fields defined as follows.

1. *Origin* defines the node which is the source of this message. Referred to as the parent, this field is characteristic of reliable route protocol. It determines the node nearest to the basic station, which is the highest in the routing table.

2. *Sender* defines the node which sent this message.

3. *Destination* defines the destination of the message which has been treated by a node; the destination can be the sink or another kind of node.

4. *Ndid* defines the node which will be identified by a nodeID which corresponds to an identifier of a node group.

5. *Type* defines the action of the different components of the system.

6. *Hop* defines a parameter of a reliable route protocol. However, it can be used for another routing protocol.

7. *Link* defines an attribute of reliable route protocol which indicates the quality of connectivity between two nodes and is very important for the definition of the routing table.

8. *Data* is composed of the different information carried by the node, including: (i) Temp (temperature parameter from the sensor board); (ii) Humidity; (iii) Pressure; (iv) GPS; (v) Conso (which defines the last energy value of the Origin node); and (vi) Activity (which determines the last CPU activity of the Origin node). *Data* also contains information about the processor activity and energy consumption of the sender node.

9. *Port* defines the output port for each message according to the DEVS rules.

We have also defined different types of message in order to describe the action which should be performed by the receiving atomic model. We list the main types of message we have defined. Even if the following list is not exhaustive, the reader will be able to discover the main defined types of message and the associated action to be performed by the receiving atomic model.

- *Router* Message for AM Net: the associated action concerns the routing information.
- *BSCollect* Message for AM Sensorboard: the associated action will collect environmental data.
- *MemCollect* Message for AM Memory: the associated action will consist of storing information.
- *ACK* Message for AM Net: the associated action will receive information.
- *WhiteFlag* Message for Net: the associated action will concern the architecture discovery signal and update of routing table.
- *DEAD* Message for all Models: the associated action will point out the fact that no energy is present.

4.2 Description of the Atomic Model AM COM

The atomic model AM COM represents communication in a node. The goal of this atomic model is to direct messages towards good nodes according to the routing table in the coupled model CM Processor.

In Figure 5, we have only depicted a link from the input port or output port to another sensor node; however, it is possible to have more links depending on connected nodes. We have defined different states in order to describe the behavior of such an device:

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016

Figure 5. Atomic model COM

- receipt state, describing the arrival of a message at Inport1 from a node or base station (BS);
- transmit state, describing the arrival of a message from a sensor node at another node or the base station BS;
- busy state, corresponding to the state of transition when a message is treated by MC processor;
- free state, describing when there is no activity in the node (when the node is listening to the channel); and
- dead state, describing that there is no battery in the sensor.

4.3 Description of the Coupled Model CM Processor

The coupled model CM Processor is depicted in Figure 6. It is one of the keys to our approach: all messages coming from an atomic model AM COM or an atomic model AM Sensorboard are necessarily taken into account by this model in order to deal with routing information. The atomic model AM Processor allows the management of all messages and components. It is difficult to represent all actions of the processor but we have been able to propose a solution using a generic approach. The defined coupled model CM Processor is the simplest representation of a generic Operating System.

We decompose the behavior of the coupled model CM Processor into three atomic models: (i) the atomic model AM Processor which represents management of the OS and Processor; (ii) AM Net which allows the management of the Network aspect; and (iii) the atomic AM Flash which is a space to store information but is also able to adapt the system to new parameters, e.g. a new type of message. These three atomic models have only three states:

1. the busy state, indicating that a model is in action;
2. the free state, indicating that a model is in sleep mode; and
3. the dead state, when there is not enough energy in the node.

When a message arrives, the atomic model AM Processor sends it to the atomic model AM Net, which describes the routing management. AM Flash is a very simple atomic model. It can stock information such as new node ID. It can also allow the definition of new types of messages, previously unknown to the system. The management of new types of messages with AM Flash allows the user to redefine new types of applications.

Atomic model AM Processor enables all messages in the model to be dealt with. We can express relative activity of a node by counting the actions performed by AM Processor.

4.4 Description of Atomic Model AM Battery

The atomic model AM Battery, as illustrated in Figure 7, is connected to all models representing the components of the sensor. Each time there is an action using some energy, the atomic model AM COM, the coupled model CM Processor and the atomic model AM Sensorboard send a message to the atomic model AM Battery.

To represent energy consumption, we use a linear model based on [17] for these first experiments. In this linear model, the battery is treated as a linear storage of current. The maximum capacity of the battery is achieved regardless of what the discharge rate is. The simple battery model allow users to determine the efficiency of the application by reporting the capacity consumed by the user. The remaining capacity C after operation duration of time t_d can be expressed:

$$C = C' - \int_{t=t_0}^{t_0+t_d} I(t)\,dt \qquad (1)$$

where C' is the previous capacity and $I(t)$ is the instantaneous current consumed by the circuit at time t. The linear model assumes that $I(t)$ will remain constant for the duration t_d, if the operation node of the circuit does not change for the duration t_d.

When a value referred to as 'size' reaches 0, a dead message is sent to all components and therefore all models enter the dead phase. All input ports are blocked and it is impossible for any model to change its state. Let us emphasize that all models have a common important state referred to as dead phase. When a sensor model enters this particular phase, it cannot act within the network any longer. This feature is essential for networking management.

Values of energy cunsumption for each component are listed in Table 2. This energy consumption depends on the duty cycle. The duty cycle is the proportion of time during which a component, device or system is operatational. Suppose a node processor operates for 1 s, then is shut off

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016

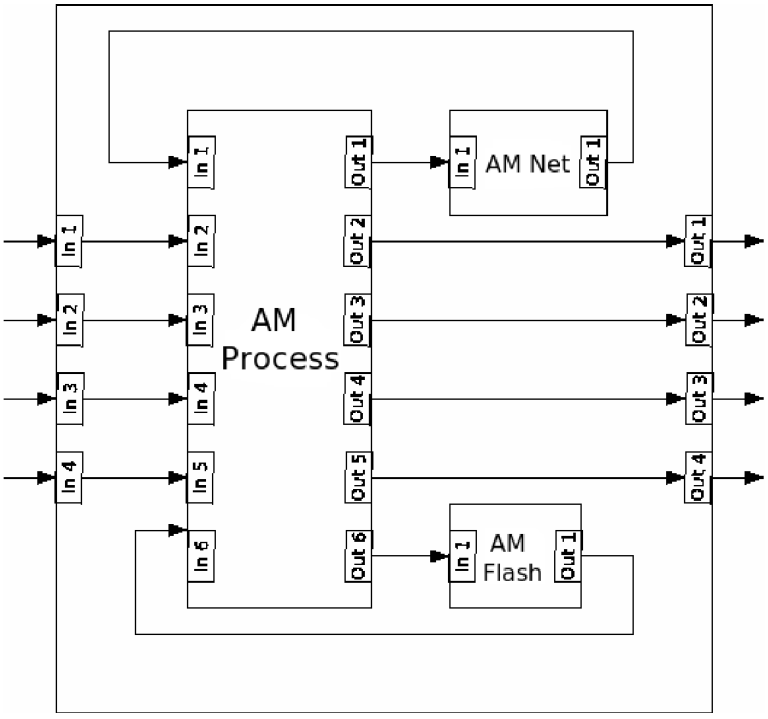


Figure 6. Coupled model Processor

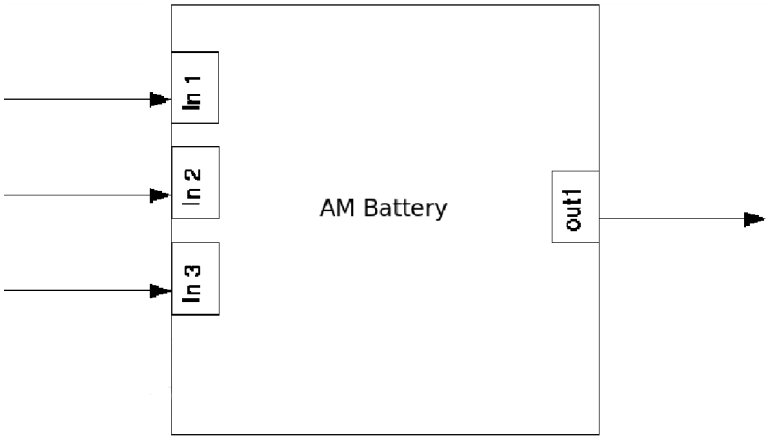


Figure 7. Atomic model Battery

for 99 s, then is run for 1 s again and so on. It runs for 1 out of 100 s and so its duty cycle is therefore 1%.

Duty cycle can be expressed

$$D = \frac{\tau}{T}$$

where D is the duty cycle, τ is the duration that the function is non-zero, and T is the period of the function. Duty cycle values for each component are listed in Table 3.

4.5 Description of Atomic Model AM Memory

The atomic model AM Memory is a simple atomic model as shown in Figure 8, which allows storage of environmental data by CM Processor. However, a coupled model

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016

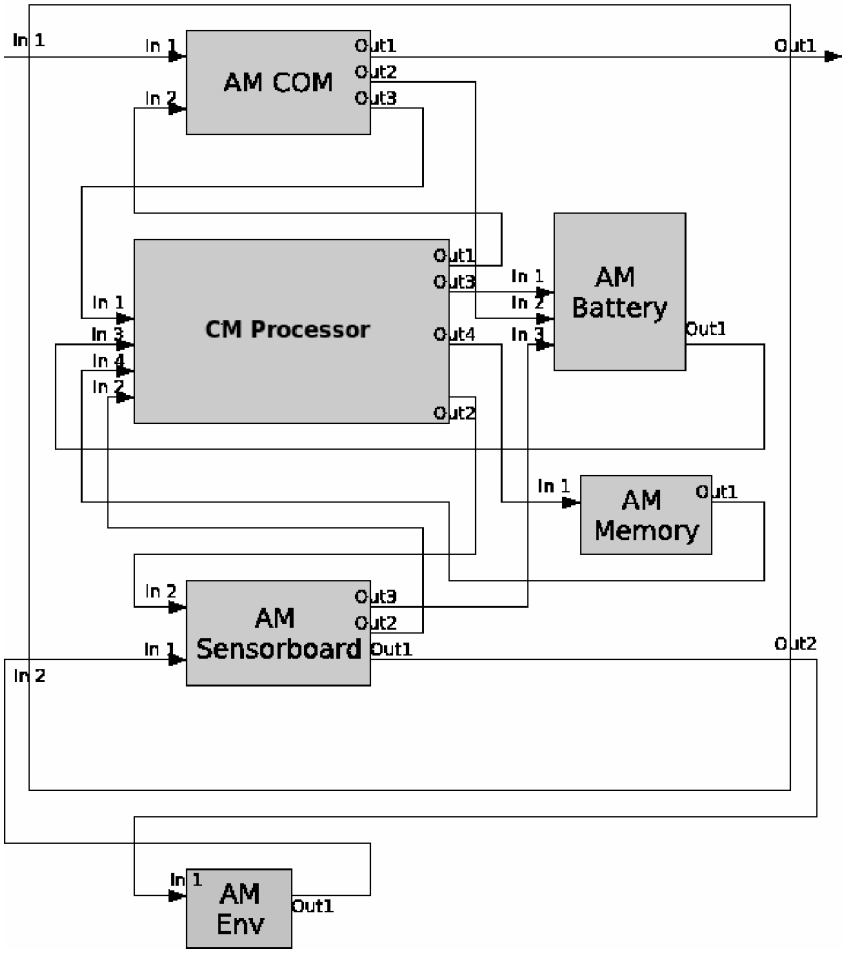


Figure 8. Coupled model Sensor

CM Processor can have different kinds of actions on atomic model AM Memory, according to the type of message. We highlight two explicit messges (MemCollect and StoreData) in Section 4.1 which are used in order to represent the actions of the processor.

4.6 Description of the Atomic Model AM Sensorboard

The goal of the atomic model AM Sensorboard is to represent interactions between the environment and sensors. It is an important part of our approach. The atomic model AM Sensorboard is connected with a special atomic model AM Env where AM Sensorboard can collect environmental data. AM Env is an external model, as depicted in Figure 9, using environmental messages to communicate with the sensor board. This interconnection represents the sensing action of nodes in an environment or a specific phenomenon (wildfire). AM Env contains different values for each environmental parameters. The atomic model AM Env allows the definition of an environmental scenario for the global network, for several groups of sensors or for each node of the WSN. To represent the variation in environmental parameters, we have implemented

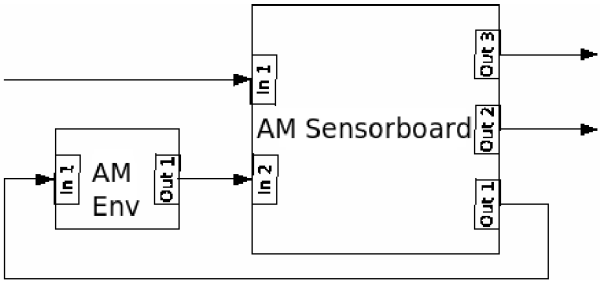


Figure 9. Atomic model Sensorboard

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016

Table 2. Energy consumption of each component [18]

System specifications			Duty cycle
Processor	Current (full operation)	8 mA	1
	Current (sleep)	8 μA	99
Radio	Current (receive)	8 mA	0.75
	Current (transmit)	12 mA	0.25
	Current (sleep)	2 μA	99
Logger memory	Write	15 mA	0
	Read	4 mA	0
	Sleep	2 μA	100
Sensor board	Current (full operation)	5 mA	1
	Current (sleep)	5 μA	99

System	Consumption (mA hr)
Processor	0.0879
Radio	0.0920
Logger memory	0.0020
Sensor board	0.0550
Total current used	0.2369

Battery capacity (mA hr)	Battery life (months)
250	1.45
1000	5.78
3000	17.35

Table 3. Duty cycle of each component for $T = 4$ s

Component	τ (s)	D (%)
Processor	0.04	1
COM (receipt/transmit)	1 (1.33)	25 (75)
Sensor board	0.04	2.5
Memory	0.01	0.25

a simple scenario with an increasing temperature on each sensor.

AM Sensorboard has five states:

1. the busy state, indicating that a model is in action;
2. the free state, indicating that a model is in sleep mode;
3. the wait state, when the model collects information from AM Env;
4. the go state, when the model sends a message to the coupled model Processor if the message doesn't contain a superior value at a fixed threshold; and
5. the dead state, when the model sends a dead message to the coupled model Processor if the message contains a superior value at a fixed threshold.

4.7 Description of Coupled Model CM Sensor

The model illustrated by Figure 8 represents coupling of the DEVS sensor model which we have defined. This definition is essential because it determines the connectivity between the model but also architectural characteristics of the future Wireless Sensor Network. This model requires two input ports, In1 and In2, and two output ports, Out1 and Out2. In1 and Out1 represent connectivity with a node. Note that the coupled model CM Sensor allows several connections with several nodes and, consequently, the number of outports and inports increases. In2 and Out2 represent the connectivity with the environment. The central role of the coupled model CM Processor is highlighted in Figure 8.

When dealing with the coupled model CM Processor, and more precisely atomic model AM Net, we work with two specific protocols called gradient-based routing protocol [19] and reliable route (or Xmesh) protocol [20].

Gradient-based routing (GBR) is a variant of directed diffusion. The key idea in GBR is to memorize the number of hops when a message is diffused through the whole network. As such, each node can calculate a parameter called the height of the node, which is the minimum number of hops to reach the BS. When multiple paths pass through a node which acts as a relay node, that relay node may combine data according to a certain function. GBR mainly uses a stochastic scheme, where a node picks one gradient at random when there are two or more hops that have the same gradient.

Xmesh protocol is more elaborate than GBR and allows the node to passively estimate the quality of the link from the other nodes by collecting statistics on packets it happens to hear, or by actively probing. Link quality is measured as the percentage of packets that arrive undamaged on a link. Link status and routing information are maintained in a neighborhood table. The goal is to have a neighborhood management algorithm that will keep a sufficient number of good neighbors in the table regardless of cell density.

To maintain this routing table, the protocol uses an algorithm based on a frequency count for each entry in the table. On insertion, a node is reinforced by incrementing its count. A new node will be inserted if there is an entry with a count of zero; otherwise, the count of all entries is decremented by 1 and the new candidate is dropped. The neighbor table contains many fields: group IDs, parent node IDs, child IDs, reception link quality and link estimator data structures.

To estimate link quality, shortest path protocol is used. For shortest path protocol, a node is a neighbor if its link quality exceeds a threshold t. Another parameter is the selection of parent node. The cost metric is used to guide

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016

routing. The cost of a node is an abstract measure of distance; it may be number of hops, expected number of transmissions or estimate of energy required to reach the sink. A neighbor is selected as a potential parent only if its cost is less than the currrent cost of a node. This protocol is able to detect and avoid cycles, detect failures of transmission and eliminate nodes in the tree if link quality worsens.

5. Implementation and Results

In order to validate the theoretical approach presented in [16] we choose to implement the described atomic and coupled models using the PythonDEVS simulator [21]. The PythonDEVS Modeling and Simulation package provides an implementation of the standard classic DEVS formalism described in Section 3. The package consists of two files: DEVS.py and simulator.py. The former provides class architecture that allows hierarchical classic DEVS models to be easily defined by sub-classing the AtomicDEVS and CoupledDEVS classes. The simulator engine (SE) is implemented in the second file. Based on the principles of simulation described in Section 3, it allows discrete-event simulation to be performed.

Even if the PythonDEVS software involves a simulation engine which offers limited means of terminating a simulation and does not provide an easy model-reinitialization possibility, we have been able to use it to efficiently test our approach. In particular, we have been able to introduce the concepts of out-of-context and in-context models because of the open source quality of the PythonDEVS package. Furthermore, by sub-classing both the atomic model Class and the coupled model Class inherent to the PythonDEVS package, we have been able to perform the simulation algorithm of PythonDEVS. This is possible by calling the methods extransition, intransition, etc. of atomic model Class and using the model's composition and connectivity, including ports and sub-models offered by the coupled model Class.

Furthermore, we implement an atomic model Generator dedicated to generate the required events in order to validate our approach. Generator has been defined according to the semantics defined by Zeigler [15].

We now describe how the proposed approach was validated. We chose to implement a benchmark network composed of eight nodes and a base station. In Figure 10, the eight nodes, the BS and the different relations between the nodes are represented. These relations represent the connectivity and capacity of communication between two nodes. If there is no connection between two nodes, it means that the nodes are much too distant to exchange information. During the simulation, all nodes periodically send messages towards the BS.

Figure 10 shows only a predefinition of relations between nodes. We choose this representation to work on the routing protocol and representation capacity of our model. During the simulation, we want the nodes to make a choice according to routing protocol rules to reach the BS.

5.1 Common Parameters of GBR and Xmesh

The sensing activity and the time of message arrival at the BS are the same for the two protocols. This fact is obvious for the sensing activity because it is only dependent upon the atomic model AM Env; however, for the time of latency, we must provide some precision. Based on the number of hops, GBR and Xmesh have the same message time of arrival at the BS.

5.1.1 Sensing Activity

According to our approach, we implemented the atomic model AM Env with a rapid increase of temperature as can be observed in the case of wildfire, for example. In Figure 11, we analyze environmental data sent periodically by the nodes and observe that the simple temperature model is clearly represented. We observe for each node a rapid increase of temperature.

5.2 Latency Time

In Figure 12, we have highlighted the time of apparition of each node in the sink table that shows the difference between a node near the sink and a more distant node. However, this time is not very important since we can see from Figure 12 that Node 7 appears after 80 s. Figure 12 shows an important shift due to the fact that there is no direct relation between nodes and the BS.

The main explanation once again is the routing protocol. As previously mentioned, a message issued from Node 7 needs to go through Node 5, Node 2 and Node 1 in order to reach the BS.

5.3 Comparison of Network Management

Here, we compare GBR and Xmesh using different parameters. In Figure 13, the architecture of the WSN is depicted. This figure shows privileged relations, i.e. the first neighbor in the routing table of each node according to the routing protocols GBR and Xmesh. Figure 14 depicts the evolution of WSN architecture for the Xmesh protocol.

We can observe that the relations between sensors are different for each protocol. Indeed, this evolution means that the routing table of Node 6 has Node 2 for first neighbor instead of Node 3. This selection of the first neighbor is made by routing rules of Xmesh according to link quality. GBR uses a stochastic scheme based on a random selection between the neighbors with the same hop number. This selection technique does not allow, in this case, a new relation definition between the nodes.

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016

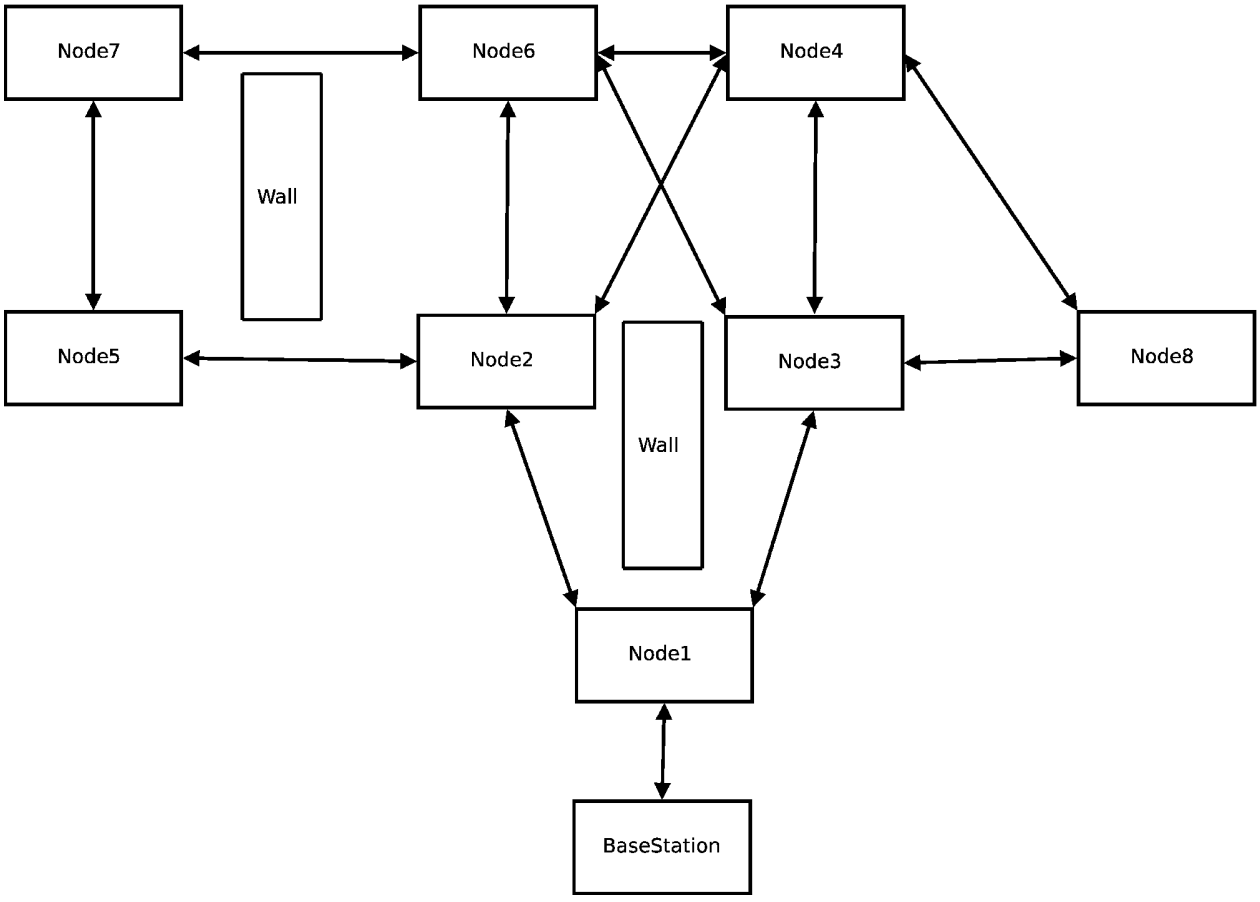


Figure 10. Network architecture for simulation

5.4 Comparison of Energy Consumption and CPU Activity

Our approach allows us to distinguish energy consumption and processor activity. To illustrate this fact, we compare the Xmesh protocol [16] and the GBR protocol according to these parameters, illustrated in Figures 15 and 16.

The events treated by each node processor model during the simulation are represented in Figure 15. In our approach, the atomic model AM Processor managed all the components and treated all actions involved in a node. Independently of the routing protocol, we can observe an important activity of Node 1 because it has a central role in the network and it is the bridge between the other nodes and the sink. This activity is the direct effect of the nodes deployment and network architecture chosen for the test represented in Figure 10.

Figures 15a and b illustrate the processor activity of GBR nodes and Xmesh nodes, respectively. We can observe that Xmesh protocol allows a more important sharing of the routing stacks than GBR protocol. Indeed, an important processor activity is a sign of the important role of the node in the routing. We can observe with GBR that Node 2 is more often sought than Node 3. Note that Xmesh reduces this problem, encouraging a better routing tasks sharing.

Figures 16a and b illustrate the energy consumption of GBR nodes and of Xmesh nodes, respectively. We can observe that the Node 2 consumption is more important in the GBR case than in the Xmesh case. These results confirm the previous conclusions, that Xmesh seems to provide a more important tasks balance in the network than GBR. Xmesh is a more complex protocol based on the number of hops to reach the base station but also has a more elaborate neighborhood selection than GBR, thus fostering a better routing tasks sharing.

6. Conclusion and Future Work

This article provides a model and simulation of the performance of a Wireless Sensor Network. This work is based

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016

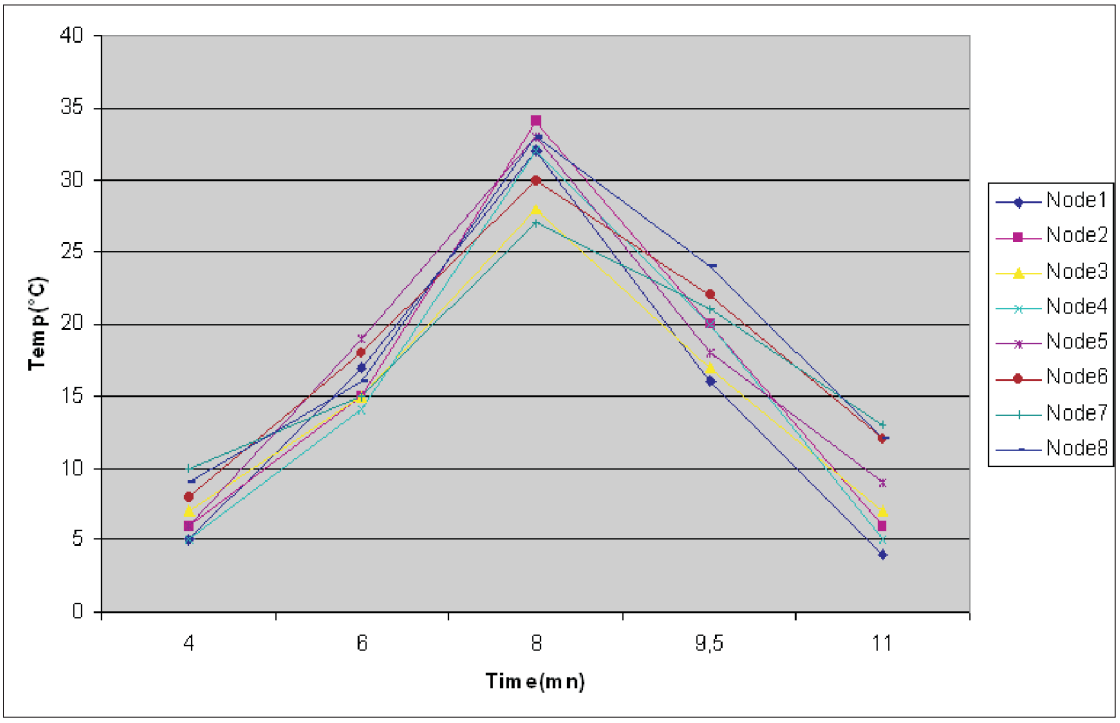


Figure 11. Environmental temperature parameter during simulation

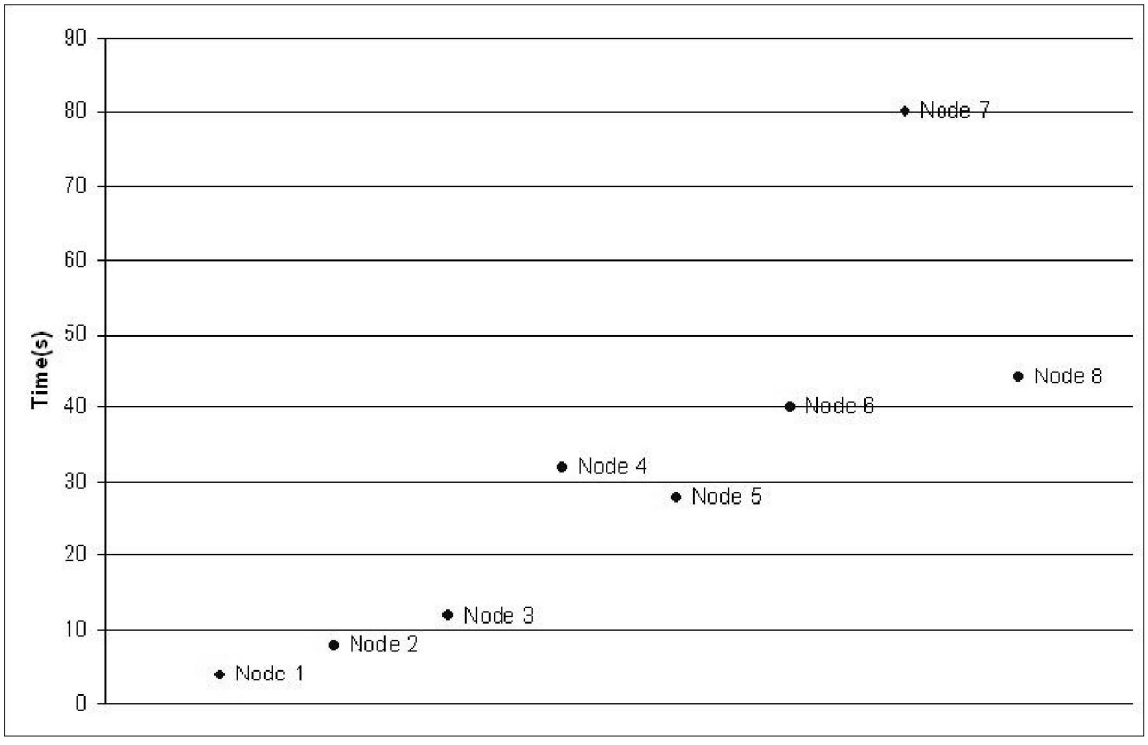


Figure 12. First apparition of node message on the sink

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016

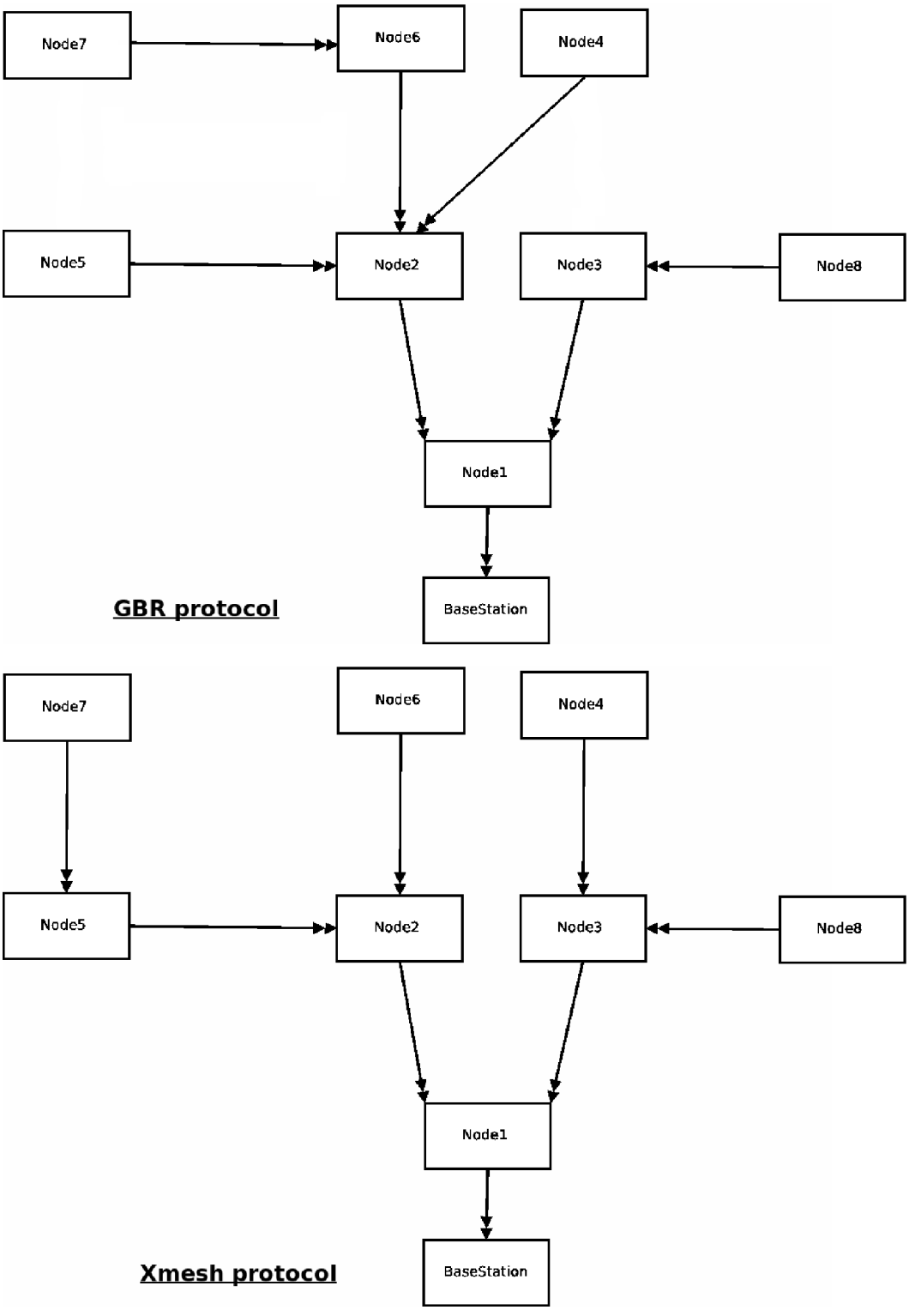


Figure 13. WSN privileged communications after 10 min of simulation

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016

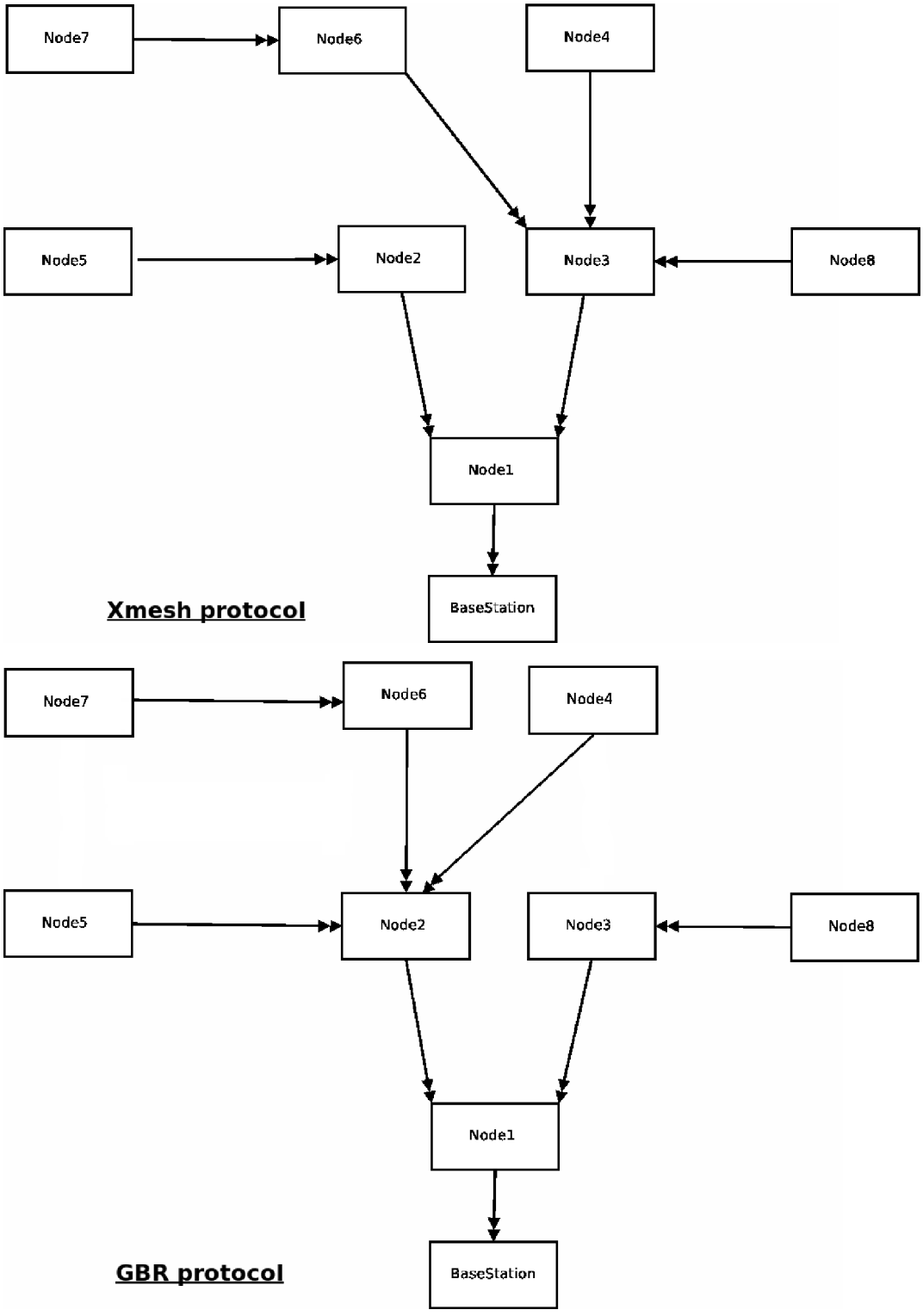
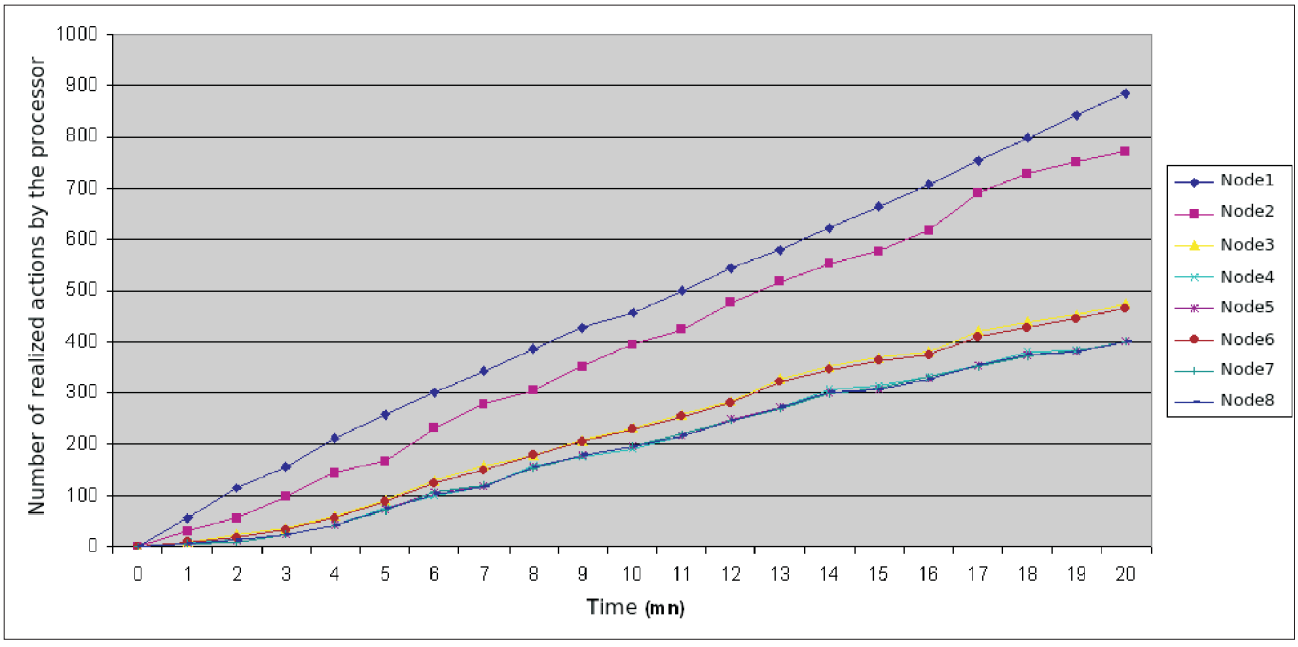
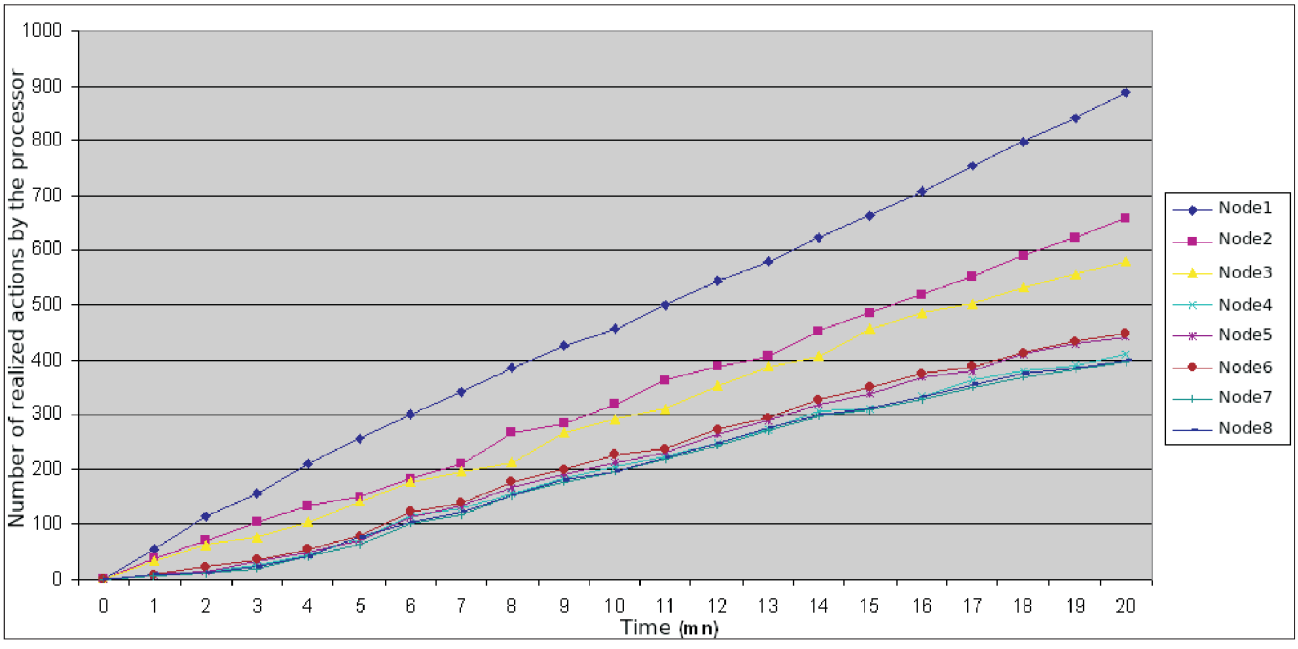


Figure 14. WSN privileged communications after 15 min of simulation

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016



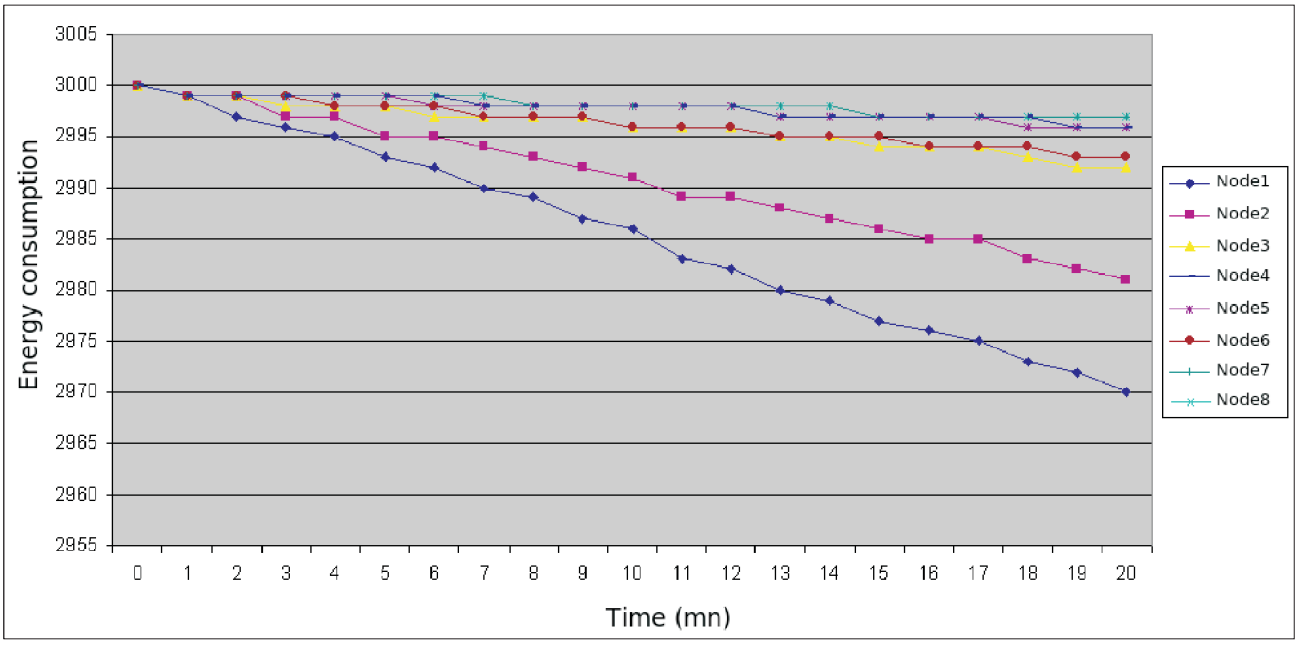
(a)



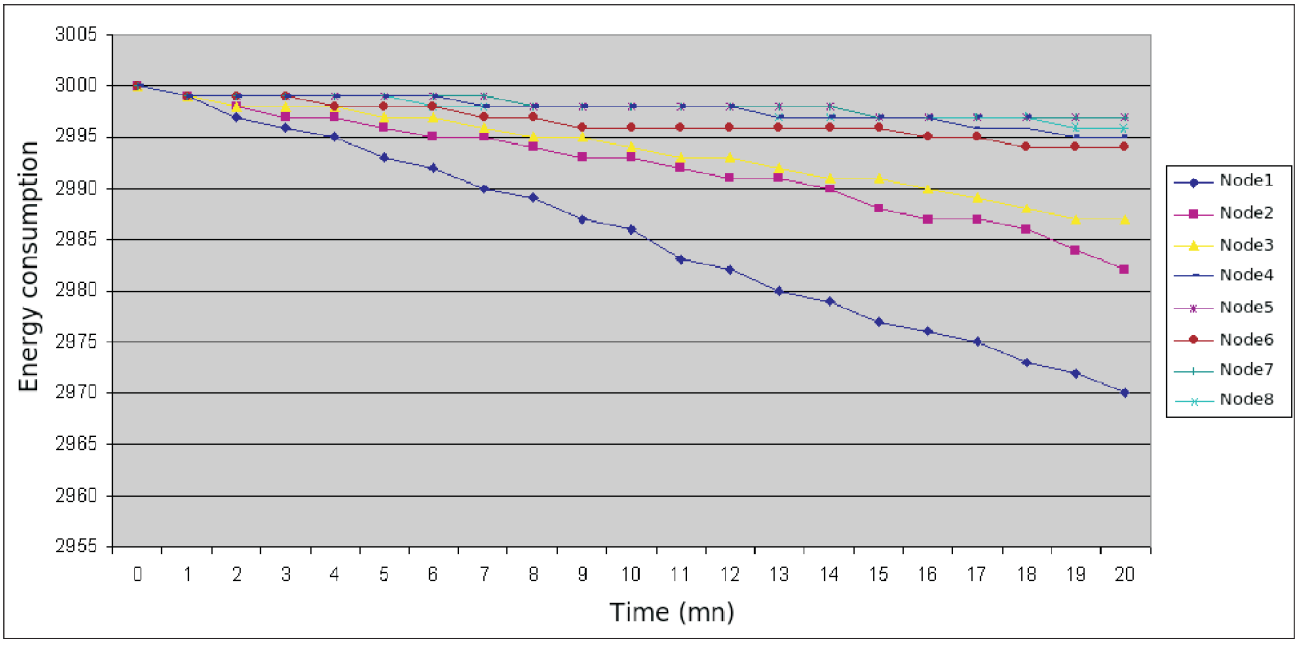
(b)

Figure 15. Relative CPU activity in the WSN

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016



(a)



(b)

Figure 16. Relative energy consumption in the WSN

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016

on DEVS formalism for the modeling and simulation of complex discrete-event systems. We have demonstrated the capacity of our approach in analyzing the evolution of a WSN architecture and the performance of a WSN according to two routing protocols, GBR and Xmesh. We have implemented the concepts presented in the paper using the PythonDEVS package and have validated our approach by providing results of a simulation of a WSN with eight nodes.

We have highlighted the routing parameters, the relative energy consumption and the CPU activity. These results allow us to demonstrate that this first approach for modeling WSN using the DEVS formalism is promising and provides a new level of node visualization. Indeed, a DEVS description of components allows us to visualize the characteristics of each component.

These results confirm both that the proposed approach is acceptable when dealing with WSN characteristics and that the DEVS formalism can efficiently model the behavior of a WSN. The modular aspect of DEVS allows us to easily change the component model of a sensor, e.g. in our example of the atomic model AM Net for the routing protocol test.

After the completion of the main components of the sensor network, an application to test the model can be created. This application is based on a DEVS simulator written in Python [21]. It comprises four packages: DEVS, ComponentsNodes, SimulationTools and WSN specification. A simulation tool called DEVS-WSN is currently under development. This simulation tool will allow us to work on particular phenomenon such as wildfire. Indeed, the atomic model AM Env and the capacity to easily study the routing protocol and the network topology will allow us to analyze the WSN behavior under forest fire conditions.

7. References

[1] Hac, A. 2003. *Wireless Sensor Designs*. Hoboken, NJ, USA: John Wiley and Sons Ltd.

[2] Akyildiz, I. F., W. Su, Y. Sankarasubramaniam, and E. Cayirci. 2002. Wireless sensor networks: a survey. *Computer Networks* 38(4), 393–422.

[3] Estrin, D., R. Govindan, J. Heidemann, and S. Kumar. 1999. Next century challenges: scalable coordination in sensor networks. *In Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking(MobiCom '99)*, pp. 263–270. Seattle, Washington, United States: ACM Press.

[4] Tilak, S., N. B. Abu-Ghazaleh, and W. Heinzelman. 2002. A taxonomy of wireless micro-sensor network models. *SIGMOBILE Mobile Computer Communication Revue* 6(2), 28–36.

[5] Khemapech, I., I. Duncan, and A. Miller. 2005. A survey of wireless sensor networks technology. *In Proceedings of the 6th Annual PostGraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNET 2005)*. Liverpool, UK.

[6] Farooq, U., B. Balya, and G.Wainer. 2004. Modelling routing in wireless ad-hoc networks using cell-devs. *In Proceedings of 2004 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2004)*, pp. 285–292. San Jose, CA, USA.

[7] Kim, T. 2006. *DEVS-NS2 Environment: An integrated tool for efficient network modeling and simulation*. Master's thesis, Univerity of Arizona.

[8] NS2. 1995. http://www.isi.edu/nsnam/ns/.

[9] Park, S., A. Savvides, and M. B. Srivastava. 2000. SensorSim: a simulation framework for sensor networks. *In Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems (MSWIM 2000)*, pp. 104–111. Boston, Massachusetts, United States: ACM Press.

[10] Polley, J., D. Blazakis, J. Mcgee, D. Rusk, and J. S. Baras. 2004. Atemu: a fine-grained sensor network simulator. *In Proceedings of IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON 2004)*, pp. 145–152. Santa Clara, CA, USA.

[11] Levis, P., N. Lee, M. Welsh, and D. Culler. 2003. TOSSIM: accurate and scalable simulation of entire tinyos applications. *In Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys'03)*, pp. 126–137. Los Angeles, CA, USA: ACM Press.

[12] Shnayder, V., M. Hempstead, B. R. Chen, G. W. Allen, and M. Welsh. 2004. Simulating the power consumption of large-scale sensor network applications. *In Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys 2004)*, pp. 188–200. Baltimore, MD, USA: ACM Press.

[13] Samper, L., F. Maraninchi, L. Mounier, and L. Mandel. 2006. GLONEMO: global and accurate formal models for the analysis of ad-hoc sensor networks. *In Proceedings of the first international conference on Integrated internet ad hoc and sensor networks (InterSense 2006)*, p. 3. Nice, France: ACM Press.

[14] Sundresh, S., W. Kim, and G. Agha. 2004. SENS: A sensor, environment and network simulator. *In Proceedings of the 37th annual symposium on Simulation (ANSS 2004)*. Washington, DC, USA: IEEE Computer Society.

[15] Zeigler, B. P. 2000. *Theory of Modeling and Simulation* (2nd edn.). London, UK: Academic Press.

[16] Antoine-Santoni, T., J. F. Santucci, E. D. Gentili, and B. Costa. 2007. Simulation and visualization method of wireless sensor network performances. *In Proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2007)*, pp. 476–483. SCS, IEEE.

[17] Park, S., A. Savvides, and M. B. Srivastava. 2001. Battery capacity measurement and analysis using lithium coin cell battery. *In Proceedings of the 2001 international symposium on Low power electronics and design (ISLPED 2001)*, pp. 382–387. New York, NY, USA: ACM Press.

[18] Crossbow-Technology. 2006. *Wireless Sensor Network Seminar*. Como, Italy, San Jose, CA, USA: Crossbow Technology.

[19] Schurgers, C. and M. Srivastava. 2001. Energy efficient routing in wireless sensor networks. *In MILCOM Proceedings on Communication for Network-Centric Operations: Creating the Information Force*. McLean, VA, USA.

[20] Woo, A., T. Tong, and D. Culler. 2003. Taming the underlying challenges of reliable multihop routing in sensor networks. *In Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys 2003)*, pp. 14–27. Los Angeles, California, USA: ACM Press.

[21] Bolduc, J. S. and H. Vangheluwe. 2001. PythonDEVS: A modeling and simulation package for classical hierarchal DEVS. Technical report, Rapport technique, MSDL, Université de McGill.

Thierry Antoine-Santoni *obtained his doctoral thesis in 2007 at the University of Corsica. His work focuses on modeling and simulation of complex systems based on the DEVS*

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016

formalism developed by BP Zeigler, Wireless Sensor Network and Bioinformatic. He has been author and co-author of many papers published in international journals or conference proceedings.

Jean-François Santucci *has been Professor in Computer Sciences at the University of Corsica since 1996. His main research interests are modeling and simulation of complex systems. He has been author or co-author of more than 100 papers published in international journals or conference proceedings. He has been the scientific manager of several European or industrial research projects. He has been the advisor or co-advisor of more than 20 PhD students and since 1998 he has been involved in the organization of more than 10 international conferences. He is conducting new interdisciplinary research involving computer sciences, archaeology and anthropology. He is currently conducting research in the archeaoastronomy field (investigating various aspects of cultural astronomy throughout Corsica and Algeria using Computer Sciences tools) and is also applying computer science approaches such as GIS or DEVS to anthropology.*

Emmanuelle de Gentili *obtained her doctoral thesis in 2002, and has been Assistant Professor at the University of Corsica since 2004. Her work focuses on modeling and simulation of complex systems based on the DEVS formalism, fuzzy logic and dynamic systems.*

Bernadette Costa *has been Professor in Electronics at the University of Corsica since 1989. Her main research interests are electronics, signal processing and acoustics. She has been author and co-author of many papers published in international journals or conference proceedings.*

Downloaded from sim.sagepub.com at PENNSYLVANIA STATE UNIV on May 9, 2016