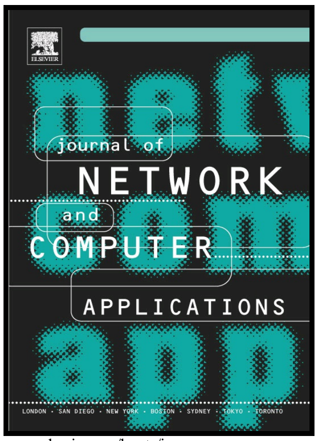

Ubiquitous Sensor Network Simulation and Emulation Environments: A Survey

Mohammad Sharif, Abolghasem Sadeghi-Niaraki

PII: S1084-8045(17)30205-9
DOI: http://dx.doi.org/10.1016/j.jnca.2017.05.009
Reference: YJNCA1922

To appear in: *Journal of Network and Computer Applications*

Received date: 20 August 2016
Revised date: 20 January 2017
Accepted date: 25 May 2017

Ubiquitous Sensor Network Simulation and Emulation Environments: A Survey

Mohammad Sharif[1], Abolghasem Sadeghi-Niaraki[1,2*]

[1.] Department of Geospatial Information Systems, Faculty of Geodesy & Geomatics Engineering, K. N. Toosi University of Technology, Tehran, Iran

[2.] Department of Geoinformatics Engineering, Inha University, Incheon, South Korea

[*]Corresponding Author: a.sadeghi@kntu.ac.ir

Abstract

Recent human effort has been directed at expanding pervasive smart environments. For this, ubiquitous computing technology is introduced to provide all users with any service, anytime, anywhere, with any device, and under any network. However, high cost, long time consumption, extensive effort, and in some cases irrevocability are the main challenges and difficulties for developing ubiquitous systems. Therefore, one solution is to initially simulate, analyze, and validate practices prior to deploying sensing and computational devices in the real world. Simulation, as a performance evaluation technique, has attracted attentions due to its speed, cost-effectiveness, repeatability, scalability, flexibility, and ease of implementation. Moreover, emulation, as a hybrid method, not only offers most simulation advantages, but also benefits from tight control of implementation, as well as a certain degree of realistic results. Both simulators and emulators are significant tools for enhancing the understanding of ubiquitous sensor networks (USNs) through testing and analyzing several scenarios prior to actual sensor placements. In this regard, this paper surveys 130 simulation and emulation environments and frameworks, which were originally designed and adapted for USN. Of these 130, the 22 that have been widely used, regularly updated, and well supported by their developers are compared based on multifarious criteria. Finally, several studies that had favorably compared the performance of simulators and/or emulators are examined. We believe the present research findings will be helpful for students and researchers to pick an appropriate simulator/emulator, and for software developers and those who are keen on producing their own environment.

Keywords: Ubiquitous Sensor Network; USN; Simulator; Simulation; Emulator; Emulation; Survey

Introduction

Information technology (IT) has been penetrating into our lives to become highly associated and interwoven with our daily activities. Computers, as user interfaces, enable individuals to connect to the cyber space and facilitate persons-to-persons and persons-to-machines interactions. Due to the rapid advancement and development in IT, cyber space has begun to resemble the real (physical) space more and more (Figure 1a), because cyber space is becoming a part of our real space (e.g., augmented reality applications). The confluence of cyber space and real space has generated a new space that has been termed *ubiquitous space* (Figure 1b). In such a smart space, which is a new generation of IT, computers are fragmented and deployed into the environment and computation is made available everywhere and anywhere through *ubiquitous computing* [1]. The word *ubiquitous* is defined as "*existing or being everywhere at the same time*" [2]. The term *ubiquitous computing* (or *ubicomp* in short) was firstly introduced by Mark Weiser [3, 4], who believed that in the near future humans will not interact with a single computer at a time. Instead, they will encounter invisible networked computers that are embedded in objects and are deployed in the environment. In other words, ubicomp is seen as a technology by which sensors interact and control the environments in an invisible manner without humans intervention [5]. All the elements are connected smartly. Computing fades into the background, rather than dominating the foreground. Ultimately, this *calm* technology will make any service accessible for all users, anytime, anywhere, with any device, and under any network. Ubicomp technology is becoming pervasive across diverse fields ranging from the military to tourism and medicine to sport.

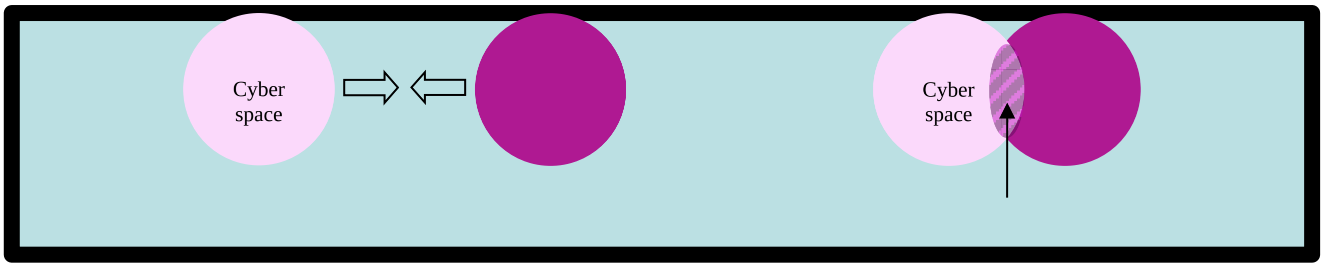

Figure 1. Information Technology (IT) spaces: (a) convergence of cyber and real spaces, (b) ubiquitous spaces produced by the confluence of cyber and real spaces.

In computer science, a network is a mixture of communication protocols and link technologies, traffic flows, and routing algorithms [6]. Networks can be in wired and wireless forms. Compared with wireless networks, wired networks have been used for several years and can transfer data more safely and securely. However, wires are one of the challenges of such networks. It is arduous to handle interwoven wires and power cords while preserving the network flexibility. Therefore, the developments of wired networks remains challenging due to wiring and rewiring bottlenecks [7]. With the rapid development of wireless technology, wireless networks are becoming widespread. Compared with traditional wireless networks, wireless sensor networks (WSNs) have more throughputs and productivity [8, 9].

The development of a ubiquitous system necessitates an infrastructure capable of supporting interrelated processing devices. Specifically, this infrastructure must be able to handle tens to thousands of static and mobile devices (known as sensor nodes or motes) where communication is performed by means of wireless transmission. Sensor nodes, with respect to their capability, are responsible for monitoring and collecting parameters [10], then processing the data locally or transmitting the data to one or more routers at ultra-high speed through ubiquitous sensor networks (USNs) [11, 12]. These nodes are physically tiny, normally cheap, and operationally low-power devices built around a microcontroller and equipped with one or more sensors, memory, radio-frequency transceiver, and a power source [13, 14]. They are deployed either stationary or movable but work unobtrusively [15].

WSN and USN differ noticeably. In WSN, sensors are spatially distributed and responsible for monitoring environmental conditions (e.g., temperature, noise, and motion), then transferring these data to central stations in wireless manner. USNs are the convergence of advanced invisible electronic devices, the Internet, and wireless networks, which not only inherit WSN features but also impose smartness into the system (e.g., the temperature separately adjusts based on individuals' contexts). Hence, WSN can be considered an infrastructure of ubiquitous computing [16]. Although USN has a broader scope, both WSN and USN may have their own meaning in different countries and applications. In this paper, these terms convey the same meaning and may be used interchangeably.

USN is the core of an ubicomp system. To have reliable, secure, and durable USN communications, a large variety of protocols is introduced in order to make use of the resources efficiently, routing the sensor packets accurately, and preserving the wireless communications effectively [17, 18]. Also, while designing USN communications, the following factors should be considered: topology of system (i.e., the arrangement of the various elements (nodes, links, etc.) of a computer network), energy consumption effects, scheduling strategies (i.e., work specified by some means is assigned to resources that complete the work), fault tolerance (i.e., continue working to a level of satisfaction in the presence of faults), data synchronization (i.e., keeping multiple copies of a dataset in coherence with one another), process synchronization (i.e., multiple processes are to join up at a certain point, in order to reach an agreement or commit to a certain sequence of action), communication range (i.e., the distance by which nodes can transfer data effectively), and coordination protocols [19]. Furthermore, given the constraints in sensor networks, such as limited resources (i.e., memory, power, quality of service, and processing ability), decentralized communications (i.e., allocation of resources, both hardware and software, to each individual node), multi-tasking (i.e., simultaneous execution of multiple applications), fault tolerance results [19], re-programmability, and security, the correlation of algorithms and protocols for these networks initially needs to be tested and evaluated. Therefore, saving time, cost, and effort requires the development of practices to be initially simulated, analyzed, and validated prior to deploying sensing and computational devices in the real world.

In this context, this research *aims to* introduce and compare the available simulators and emulators environments and frameworks for USN applications. The rest of this paper is organized as follows. In Section 2, we discuss the performance evaluation techniques related to USN, as well as their corresponding merits and demerits. Section 3 describes and compares the USN operating systems (OSs). Related studies in reviewing USN simulation and emulation environments are comprehensively presented in Section 4. In Section 5, an overview and classification of 130 USN simulators' and emulators' environments and frameworks are provided. Section 6 compares several USN simulators and emulators based on multifarious criteria and follows with pros and cons of the selected ones in tabular format. In Section 7, performance results and conclusions of applying simulators and emulators from previous studies are addressed. Potential future works and open issues related to USN simulation and/or emulation are discussed in Section 8. Finally, we summarize and conclude with final remarks in Section 9. All the acronyms and abbreviations used in this paper along with their definitions are provided in Table 1.

Table 1 List of acronyms/abbreviations and corresponding definitions.

Acronym/Abbreviation	Definition
IT	Information Technology
WSN	Wireless Sensor Network
USN	Ubiquitous Sensor Network
OS	Operating System
ABM	Agent-based Model
GUI	Graphical User Interface
GNU GPL	Gnu's Not UNIX General Public License
BSD	Berkeley Software Distribution
CRSN	Cognitive Radio Sensor Network
API	Application Programming Interface
IoT	Internet of Things
CS	Cyber Space
RS	Real Space
a	Academic

r	Research
c	Commercial
G	Generic Network Simulator
C	Code Level Simulator
F	Firmware Level Simulator
A	Algorithm Level Simulator
P	Packet Level Simulator
I	Instruction Level Simulator

USN performance evaluation techniques

Several techniques have been introduced for performance evaluation of protocols and algorithms in USN, including analytical modeling, simulation, emulation, testbed, and real-world experimentation [20]. Analytical models are a set of equations that represent the performance of a system. Although analytical models simplify the modeling procedure, they cannot accurately represent the inherent complexity of sensor networks [21]. Simulation has been cited as the most frequent and effective method for designing and developing network protocols and algorithms [20]. By using simulators various scenarios of the real environment can be modeled. Also, they provide the possibility of testing and debugging protocols at any stage of design. Emulation, as a hybrid method, is a combination of hardware and software components accompanying simulation possibilities for network modeling [10]. Emulators use firmware as well as hardware to execute simulations in laboratory conditions. Since emulators can be utilized in real environments, they potentially perform precisely in comparison to simulators [7]. Physical testbeds are frameworks for real implementation of protocols and algorithms. Testbeds not only allow remote configuration, running, and monitoring experiments but also support model, protocol, and algorithm evaluation. They have bridged the gap between simulation and deployment of real devices [10]. A comprehensive survey of current testbeds can be found in [22-27]. Real-world experimentation allows feasible and actual sensor deployment practices. All the functions are set in the reality and no incorrect or inaccurate presumption is made. This is the ultimate stage of the validation of protocols and algorithms [7]. Each of the aforementioned techniques has its own pros and cons, which are summarized in Table 2.

Table 2. Pros and cons of USN performance evaluation techniques.

Performance evaluation techniques	Pros	Cons
Analytical model	-Low cost -Provides quick insight -Provides initial evaluation	-Deduced results are not precise in terms of consumed energy, memory, processing power, sheer number, unattended operation, and harsh environments of sensor nodes

Simulator	-Fast -Low cost -Ease of implementation -Repeatable -Supports tight controlling -Scalable (supports large number of nodes) -Supports dynamic and flexible modeling -Supports heterogeneous operating systems and programming languages	-Software may contain oversimplified protocols -May not generate accurate result as real implementation -Considers high degree of abstraction
Emulator	-Repeatable -Supports tight controlling -Provides certain degree of realism	-Cost per tested node is high -Technical scalability bounds -Low speed -Limited scalability -Platform dependence
Testbed	-Demonstrate applicability of protocols in real environments -Allows to validate prototypes -Efficient in incrementing potentially long-lived experiments -Bridges the gap between simulation and deployment of real devices	-Complex -Costly -Time consuming -Limited scalability -Difficult to repeat experiments -Not replicable for hazardous environments
Real experiment	-Accurate and reliable results -No hypotheses and abstraction of reality	-High cost of software, hardware, and manpower -Difficult to repeat experiments -Resource constraints -Limited scalability -Limited tight control

USN performance evaluation techniques range from purely software-based to solely hardware-based techniques. To clarify the nature of these techniques, Figure 2 depicts the contribution of each in terms of the proportion of virtual and real spaces which they use. In this regard, analytical models and simulators only perform in virtual space, and no physical deployment is implemented in real space. For emulators and testbeds this sounds different. These techniques apportion their throughput to cyber and real spaces. In the former, cyber space has a much great portion, while in the latter the majority of the implementation is dedicated to real space. Real deployment, as the latest evaluation technique, fully concentrates on real space. All the implementation and manipulation in this technique go through physical deployment [28]. Considering all the positive and negative aspects of USN performance evaluation techniques, this research aims to investigate simulation and emulation concepts and environments. This will not only enable us to assess the nature, ability, and productivity of USN simulators and emulators but also allows evaluating techniques that are performed purely in cyber space and those in a mixture of cyber and real spaces.

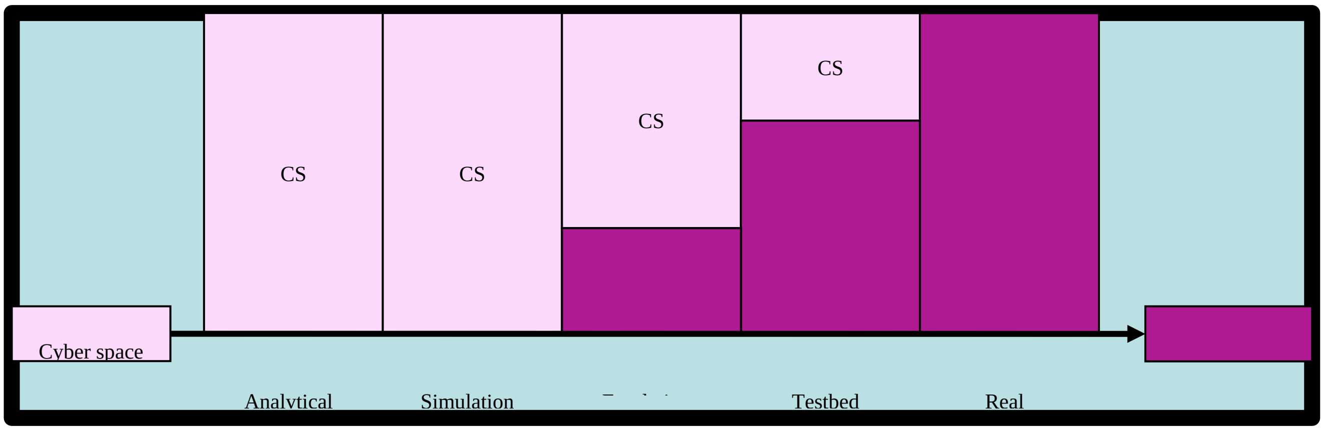

Figure 2. The proportion of performance evaluation techniques from cyber space (CS) to real space (RS).

The difference between agent-based modeling and ubiquitous computing can be contentious. On the one hand, agent-based models (ABMs) have been used diversely to study the complex interaction of entities of the real world [29]. Analytical models and simulators are the prominent performance evaluation techniques used by ABMs. In other words, ABMs are summarized in algorithms within simulators through virtual space. By achieving a certain degree of confidence from agent-based modeling, physical practices may be implemented into real space. ABMs, however, suffer from the deficiencies of analytical models and simulators such as over simplification and high level of abstraction, to name a few. On the other hand, ubiquitous computing takes place everywhere and is not limited to boundaries. In contrast to ABMs, ubiquitous computing is accomplished by real deployment of pervasive computing devices in real space. However, agent-based modeling can be a prerequisite step for ubiquitous computing. To clarify these terms, Figure 3 demonstrates the stage of ABM and ubiquitous computing by means of performance evaluation techniques.

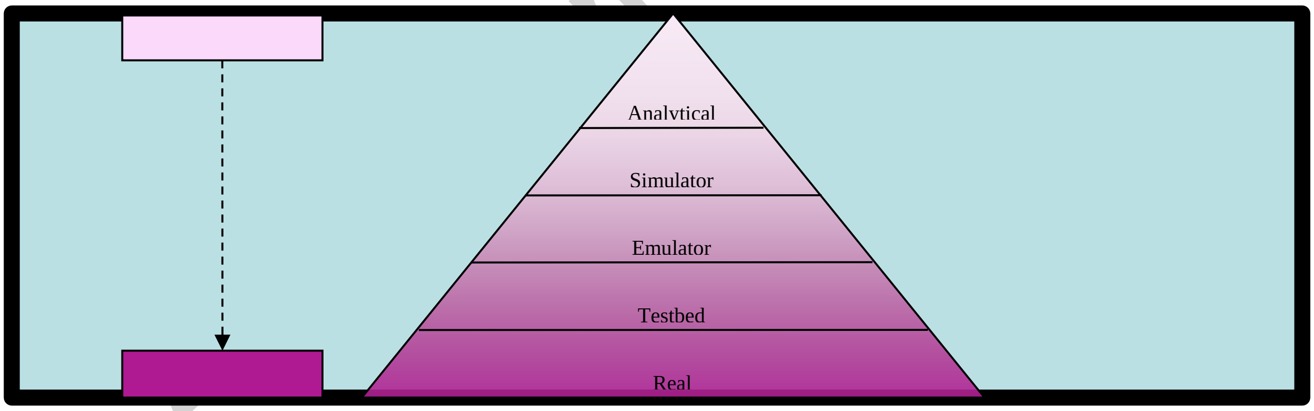

Figure 3. From agent-based modeling to ubiquitous computing.

There is a tradeoff between ABM and simulators. By ABM, a set of rules is defined at agent level and their interaction is modeled explicitly. Simulators, as experimental tools, typically convey a general meaning and cover broader domain. They are more or less dependent on the predefined rules in the software. However, the rules at ABM can be imported to simulators in order to determine the behavior of the whole system at a global level. This procedure is known as multi-agent simulation [30]. Nevertheless, simulators cannot handle agents and their corresponding rules.

Simulation

The imitation of the real-world’s conditions and processes in the course of time is known as *simulation*. By simulation, the system behavior can be characterized and analyzed, what-if questions can be raised, and systems with close similarity to real conditions can be designed. Significant information regarding the feasibility, productivity, and efficiency of a system can be assessed by simulation prior to real deployment of actual implementation [31]. Normally, to carry out a simulation, a model needs to be developed. Such a model demonstrates the main properties, characteristics, and treatment of the desired system/process. The model represents the system itself, whilst the operation of the system in the course of time is shown by simulation. However, it is not trivial to derive a trustworthy conclusion from a simulation result [32]. Diverse steps exist during a simulation and may vary with respect to the purpose of simulation. These steps are not necessarily sequential and can be applied in non-consecutive manner. Nevertheless, evaluating the performance of the model requires cyclic revision and thorough evaluation of the functionality of the simulation. Figure 4 outlines the simulation process and steps as they are concisely described in [33].

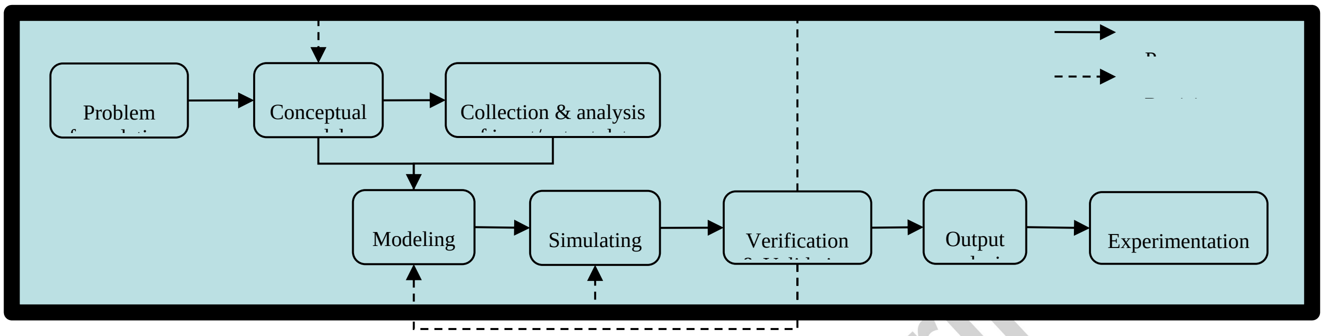

Figure 4. Phases in simulation studying (extending the textual descriptions of [33]).

Simulation types

Three types of simulation have been mentioned in computer science literature: *Monte Carlo simulation*, *Trace-driven simulation*, and *Discrete-event simulation*. Monte Carlo simulation is a static simulation or one without a time axis. It is used for modeling probabilistic events whose characteristics do not vary over time. Also, Monte Carlo simulation is utilized to appraise non-probabilistic expressions by making use of probabilistic approaches. Trace-driven simulation uses a trace as an input in the process of simulation. A trace is defined as a time-ordered history of phenomena in a real system. In general, Trace-driven simulation is used in analyzing or tuning resource management algorithms. Discrete-event simulation, in contrast to continuous-event simulation, uses a discrete-state model of the system for simulation and is used due to the variable system state which is described by the number of jobs at various devices. Time in discrete-event simulation can be discrete or continuous [34]. The last two simulation types are widely used in USN due to their high performance and scalability (i.e., possible number of static and mobile sensors).

Simulation execution

Simulators either run via *synchronous* or *asynchronous* modes. *Synchronous simulation* [35], on the one hand, is the simplest simulation method and is a round-based technique: Firstly, the global time increases by one unit via the framework. Secondly, the nodes move with respect to their mobility models and the connections are updated according to the connectivity model. Finally, this procedure iterates over nodes [36]. Synchronous simulation has positive aspects, including ease of implementation, performance predictability, and low overhead [37]. However, it tends to suffer from weak load balancing and communication costs due to the synchronization steps between rounds. In brief, synchronous simulation is appropriate for simulations with short computational granularities and great round parallelism [38]. On the other hand, *asynchronous simulation* is highly based on events. A number of message and timer events are aligned in time intervals which should take place in order. The events are picked and executed via the framework repeatedly [36]. Conservative simulation and optimistic simulation are two types of asynchronous simulation. Comparing these two simulation modes, synchronous simulation runs slower than asynchronous simulation mainly because synchronous simulation meets all the nodes including the ones that are nonfunctional. This condition is not applied for asynchronous simulation. In this mode, only the messages and timer events are processed and unnecessary rounds are not implemented. Asynchronous simulation mode does not support node movement because the continuity of nodes mobility cannot be characterized as events [36].

USN simulation

In the USN domain, simulation is one of the most prevailing appraisal procedures for the progression of wireless network protocols and communication frameworks, and for assessing the available ones in different scenarios [39]. The simulators designed for USN purposes are commonly designed to consider the development constraints (e.g., node and communication). Based on the nature of constraints, simulation tools are classified into (1) *oriented network*, and (2) *oriented node* classes [40]. Oriented networks concentrate on the wireless networks behavior and the protocol stack of the operation. These simulators are initially designed for network simulation and then extended for USN purposes. Examples of this class of simulator are OMNet++, NS-2, and J-Sim. Oriented node, as the second class of simulators, concentrates on the functionality of a single node that contains simple communication models. These simulators are particular to targeted nodes and their OSs. Furthermore, these simulation tools are able to determine the compatibility of a node with an application. Examples of this class of simulator are TOSSIM, ATEMU, and SENS. Two common aspects are considered by these two classes: (1) the correctness of the simulation models, and (2) the suitability of a particular tool to implement the model. Generally, a USN simulator contains multiple modules, including [41]:

Node is a device composed of both hardware and software in USN. Nodes components are actuator, sensor, processor, transceiver, network protocol, energy resource, and application.

Event represents substantial functionalities including the time in which an event takes place.

Medium module enables nodes to transmit signals and informs the nodes regarding affective signals.

Environment module enables the propagation of physical phenomena, such as humidity, sound, temperature, and light to be modeled.

Transceiver hardware determines the state of each sensor node (i.e., sleep, standby, receive, and transmit) as well as nodes power consumption.

Physical Protocol is known as the lowest layer of a network stack. It enables services such as transceivers state alteration and packet transmitting and receiving.

MAC Protocol resides above the physical protocol. It is normally installed on the node processor software. MAC protocol enables services such as alteration of MAC layer state and defining protocol parameters.

Routing Protocol is located above the MAC protocol. It enables messages to be routed between network hobs.

Application Layer lies on the top of the network stack. It implements an USN application through connecting with lower layers, sensors, and actuators.

USN simulator categories

Simulation can be applied at various abstraction levels, from generic simulation, where only the most important features are modeled, to highly detailed simulations, where particular aspects are represented. A research contribution [42] has categorized the simulators based on the level of abstraction.

Generic Network Simulators concentrate on network simulation more than node simulation. High level languages are used for writing simulation applications, which is far removed from real sensor language. Also, the same programming language is used for applications and protocol codes. Most of the network simulators provide simulation of network stack, MAC protocol, and radio medium. Generic network simulators are effective for assessing new communication protocols. However, they are less operational for interoperability evaluation or exploring software bugs.

Code Level Simulators make use of similar codes as are utilized in actual sensor network nodes. So, they enable network stacks executions which are presented for a particular OS. Code level simulators not only enable the simulation of radio medium but also provide sensor simulation. They are effective in the detection of software bugs (e.g., deployable code or logical error), but they are not appropriate for hardware ones (e.g., CPU architecture or timing).

Firmware Level Simulators consider both sensor node emulation and firmware that run in the actual sensor network. Firmware level simulators enable detailed simulation and produce accurate implementation results. Furthermore, they facilitate radio medium simulation in addition to microprocessor and radio chip emulation. Because of detail simulation, firmware level simulators execution times are higher than those of generic network or code level simulators.

Another study [38] has classified the simulators into three major categories based on the level of complexity.

Algorithm Level Simulators consider the logic, data structure, and presentation of algorithms. Algorithm level simulators concentrate on graph data structure to represent nodes connections rather than detailed communication modeling. They enable large network simulation but with no simple MAC layer protocol.

Packet Level Simulators execute the physical layer and data link into the network stack. Thus, they provide MAC protocols and radio models to be implemented, which are the ones that are feasible for propagation, collision, fading, and noise and wave diffraction.

Instruction Level Simulators, also named emulators, provide CPU execution modeling at the level of instructions.

Requirements for USN simulation

Given the multifarious features of USN in decentralized communication, such as multitasking, heterogeneity, numerous sensor nodes, and limited resources, the design and development of a simulator is a challenging issue [43]. In this context, six key factors for USN simulation tools should be taken into consideration.

Fidelity focuses on the faithfulness of simulation as well as prediction of system behavior. In this regard, for radio channels, physical environment, node system, and accurate models need to be developed.

Scalability represents the supported number and density of sensor nodes by a USN simulator. As USN applications require the deployment of many sensor nodes, higher scalability of a simulator is an advantage.

Energy aware is a critical feature in USN simulators. Since sensor nodes have restricted resources of energy (i.e., battery or solar cells), power consumption and timing information need to be modeled accurately via simulators prior to the real deployment of sensor nodes.

Extensibility enables users to modify the available modules or import new ones to the simulator. A user-friendly interface with high modularity aids users to add or alter the functionalities.

Heterogeneity support enables the integration of a variety of multifarious elements in USN simulation tools. This includes modeling of various nodes and their interconnections.

Graphical User Interface (GUI) facilitates the implementation of the network topology and the composition of modules. It can also speed up debugging, tracing, and visualization of the simulation results.

Simulation criteria assessment

There are multifarious criteria for assessing a simulator. Key properties such as reusability and availability, performance and scalability, support for rich-semantics scripting languages to define experiments and process results, and graphical, debugging, and trace support should be present in a good simulator [8]. Also, there are diverse critical features for simulators which are categorized into input, processing, output, support, and cost groups [44]. Each category comprises several criteria that are outlined and extended in Table 2. Based on the design goals, architecture, and applications abstraction level, a combination of these features can be present in a simulator.

Table 3. Features of simulation software.

Input features	Processing features	Output features	Environment features	Cost features

-Interface to other software -Input data analysis capability -Portability -Syntax -Input flexibility -Modeling flexibility -Modeling conciseness	-Execution speed -Model size -Material handling -Random variant generators -Reset -Independent replications -Attributes -Global variables -Programming -Conditional routing -Rare event simulation	-Standardized reports -Customized reports -Confidence intervals -Business graphics -File creation -Tracing capability -Data base maintenance -Post processing and statistical analysis	-Ease of use -Ease of learning -Quality of documentation -Animation capability • Ease of development • Quality of picture • Smoothness of movement • Portability for remote viewing • CAD interface -On-line help -On-line tutorial -Customer support • Training • Technical support • Update and enhancement	-Hardware requirement -Time spent learning to use the software -Time required for building models

Emulation

The tools which comprise of software and hardware to perform the simulation are typically known as *emulators*. In an emulator, the actual hardware (e.g., motes), beside simulated components (e.g., links and traffic), aims to provide realistic performance for USN applications. The emulator usually has high scalability for simultaneously emulating several sensor nodes. Comparing to simulators, emulators are implemented in real sensor nodes and run real application codes, which improves their performance precision. Emulators are appropriate for timing interactions among sensor nodes as well as for fine tuning network level and sensor algorithms [19].

In a research contribution [10], emulators are categorized into *physical layer* and *MAC layer* classes. For physical layer emulators, a real system is comprised of all the network layers except the physical layer. These emulators rip the emitted radio signals via nodes wireless interfaces in order to experience the effects that radio waves may face in reality. On the contrary, inverse physical layer emulators act the other way round, i.e., the overhead parts of the network group are simulated and packets using real hardware are transmitted. For the MAC layer emulator, a real system is comprised of all the network layers except the physical layer and the MAC layer.

Taxonomy for USN simulation and emulation tools

Importing node models into network simulators has been an evaluation approach of USN simulators. Two types of node model have been introduced as (1) *simulators node models*, and (2) *node emulators*. The latter relates to instruction level simulators of the nodes microcontrollers, and is comprised of sensors and transceivers extensions as well as diverse peripheral models. *Node emulators* enable modeling of the network and inserting node models into network simulators [43]. In this regard, USN evaluation is categorized into the following four classes.

Network simulators with node models focus on discrete-event timing, radio medium, network modeling, and more or less the sensor node sleep duty cycles.

Network simulators with node emulators benefit from the merits of both the network simulators and node emulators. A detailed network model can be achieved through network simulators. Also, accurate timing information of the tools can be gained by the node emulator.

Node system simulator with network models operates frequently at the system level via hardware description languages, such as SystemC. Such languages enable node hardware modeling in diverse abstraction levels with various details, such a system level, transaction level, and register transfer level.

Node emulators with network models can execute the application code directly. The node emulators can be classified as (1) instruction set simulators for special microcontrollers, and (2) emulators designed to emulate the execution of the application code of an OS.

Simulation and emulation output

Simulation and emulation outcomes can be represented as graphs, text files, and animations of a trace file. Graphs facilitate comparison among multiple protocols can be accomplished. Graphs can demonstrate the variation in packet delivery amount, network delay and throughput, and several other parameters for network performance assessment. The output text files can be inputs for other simulators or programs. Ultimately, every event that happens in the simulation process can be recorded via a trace file [45].

USN Operating Systems

As stated before, USN is comprised of several tiny sensor nodes that communicate through wireless networks. The components of sensor nodes hardware such as physical sensor, microprocessor/microcontroller, memory, radio transceiver, and battery need to be operated in orderly and controlled manner. This process is conducted via an OS. Thus, each sensor node requires an OS for controlling the hardware, providing hardware abstraction to application software, and reducing the gap between applications and the underlying hardware [46]. In other words, OS acts as a resource manager for allocating resources correctly and effectively without any conflict [47]. For USN purposes, OSs must provide basic functionalities, efficient power management mechanisms, field reprogramming mechanisms, and a configurable communication stack, as well as the ability to abstract heterogeneous sensing hardware in a uniform fashion and operate with limited resources [48].

[49] presented a classification framework for USN OSs based on their important features, i.e., architecture, execution model, reprogramming, scheduling, and power management. In addition, it proposed adequate OSs for various classes of USN applications. [50] reviewed the architecture and performance analysis of five USN OSs: TinyOS, Contiki, Mantis OS, SOS, and Microsoft .NET Micro. [51] addressed the major challenges in designing OSs and reviewed some important features of TinyOS, Contiki, Mantis OS, SOS, Nano-RK, RETOS, and LiteOS OSs. Over the past years, a variety of OSs has been introduced to facilitate developing USN applications. A list of the identified ones is presented in alphabetical order in Figure 5. Reviewing all of them is beyond the scope of this research, but for further information please refer to [49]. Aside from the mentioned OSs, several studies have tried to enhance OSs capabilities in diverse dimensions, for instance, improving OS reliability (e.g., t-kernel, Harbor, and Neutron), enabling real-time support (e.g., FIT), extending the programming model (e.g., protothreads and TOSThreads), and enabling reprogramming support (e.g., Deluge, FlexCup, Stream, and Elon) [52].

- AmbiCompVM	- Embedded Linux	- LiteOS	- NanoVM	- SenSpire OS
- AVRX	- EMERALDS	- µCOS	- OSPM	- SensorOS
- Bertha	- EYES	- MagnetOS	- OSSTAR	- SmartOS

- BTnutOS or NutOS	- FreeRTOS	- Mantis OS	- ParticleVM	- Squawk VM
- Contiki	- GenOS	- Maté	- PeerOS	- SOS
- CORMOS	- Jallad	- Microsoft .NET Micro	- PicOS	- T2
- CustomOS	- kOS	- MoteWorks	- Pixie OS	- TinyOS
- CVM	- KVM	- Nano-QPlus	- RETOS	- YATOS
- DCOS	- LORIEN	- NanoRK	- SenOS	- VMSTAR

Figure 5. List of sensor network OSs.

Related works

Over the last decade or so, a plethora of researches has exploited USN simulators and emulators, demonstrating the utility and significance of these tools in USN applications. Consequently, a considerable body of researches has *specifically* and *generally* overviewed, compared, and evaluated different aspects of the USN simulation and emulation environments/frameworks.

From the specific point of view, [53] investigated the energy-aware suitability of a number of USN simulators, and subsequently proposed a novel structure for simulating energy-aware USNs. [54] introduced and assessed the coverage and connectivity features of popular USN simulation tools. [55] provided background on a number of different sensor web simulation tools along with the advantages and the drawbacks of each. Accordingly, they proposed an evaluation methodology in order to assess the capabilities of each simulation tool. Although the significance of such specialized investigations is indubitable, the outcomes cannot be extended to all the features of that distinguished tool. In other words, USN applications are normally comprised of a set of stages and implementations which a simulator/emulator should be able to handle. The strength of a simulator/emulator in specific feature does not guarantee that other features perform well too. Therefore, several general aspects of simulators/emulators require to be evaluated in parallel.

From the general standpoint, the majority of published survey papers have investigated USN simulation and emulation environments/frameworks either in *quantitative* or *qualitative* manners, and rarely a combination of these two can be seen in literature. By the quantitative studies, the majority of researches have reviewed a large number of simulators and/or emulators at naive levels by providing short descriptions and general overviews of the tools so far. For example, [56] glimpsed 63 simulators, 14 emulators, 19 data visualization tools, 46 testbeds, 26 debugging tools, 10 code-updating tools, and 8 network monitors in USN. [57] presented the state-of-the-art, main features, and the GUI snapshots of the 35 widely used USN simulation and emulation environments. [58] listed 74 USN simulators and emulators accompanying their features and properties. Although such studies are treated as overview articles, none of them compared and evaluated simulation and/or emulation tools in depth.

By the qualitative studies, a few research contributions have studied a limited number of simulators and/or emulators by providing meticulous details and specific features of those tools. For example, [59] deeply compared the NS-2 simulator and the TOSSIM emulator in terms of models, visualization tool, architecture, event scheduler, and components. [60] evaluated the interface, accessibility and user support, availability of USNs modules, extensibility, and scalability of seven (i.e., NS-2, OMNeT++, GloMoSim, OPNET, SENSE, TOSSIM, GTSNetS) simulation and emulation environments. [61] effort was toward examining the antenna setting, radio propagation, noise, medium access control, topology, and energy consumption modeling just in four (i.e., Castalia, MiXiM, TOSSIM, WSNet) USN simulators and emulators. It is evident from the studies alike the ones abovementioned that particular features of limited number of tools have been normally assessed and there is no necessity that two studies evaluate identical features of one simulator/emulator. For example, both [59] and [60] studied NS-2, albeit with different criteria. [60] and [61] studied TOSSIM with almost distinct properties.

Table 4 presents a chronological overview of both quantitative and qualitative studies over the past decade. This summery is conducted by reviewing the publicly available and published documents including journal articles, book chapters, conference proceedings, theses, and technical reports. Among the preceding contributions related to the evaluation of simulators and emulators for USN, none of them have profoundly focused on all the present developed/extended tools so far, very few of them (e.g., [62]) have comprehensively evaluated the prominent tools based on various criteria, and no structured classification has been suggested for these tools in the literature. Moreover, to the best of our knowledge, there is no research contribution that has studied the performance assessment parameters of simulators and emulators in USN scenarios. Furthermore, a few new tools have recently released and some of the well-known traditional simulators/emulators have been developed since past few years that should be examined. Therefore, there is an overriding need to fill these gaps in a new survey article.

This survey is different from the existing reviews in three salient aspects. Firstly, this survey expands its investigation to all of the (founded) USN simulation and emulation environments/frameworks along with their derivatives and extensions produced so far. Specifically, this article is not only focusing on the quantitative aspect of simulation and emulation environments (i.e., 130 tools) but also is qualitatively assessing the ones which have been widely used, regularly updated, and well supported by their developers based on multifarious criteria. It also suggests a categorization for these tools on general- and specific-purpose basis. Secondly, this survey provides a general picture on the-state-of-the-art evaluation criteria for both simulators and emulators. This consequently determines which prominent tool is adequate for what kind of purpose (i.e., academic, research, or commercial). Thirdly, this survey proposes a number of performance assessment parameters for simulators and emulators in ubiquitous simulation and emulation scenarios.

Table 4. Contribution of the reviewed literature in USN simulators and emulators.

Refer ence	ear	Simulators and/or Emulators	Description
[63]	005	NS-2, SENSE, GloMoSim, SENS, SensorSim, ATEMU, OMNeT++, Prowler, J-Sim, Shawn, TOSSIM, OPNET, TOSSF	Properties
[8]	005	NS-2, OMNeT++, J-Sim, NCTUns2.0, JiST/SWANS, GloMoSim, SSFNeT, Ptolemy II, TOSSIM, EmStar/EmSim/EmTOS, SENS, ATEMU, Prowler/JProwler, SNAP	Overview and implementation issues
[59]	005	NS-2, TOSSIM	Models, visualization, architecture, components
[64]	006	SSF, SWANS, J-Sim, NCTUns2.0, NS-2, OMNeT++, Ptolemy, SNAP, ATEMU, EmStar, TOSSIM	Models, type of visualization
[65]	007	NS-2, J-Sim, SENSE	USN application in medicine, overview, and comparison
[55]	008	NS-2, OPNET, OMNeT++, J-Sim, NCTUns, JiST/SWANS, GloMoSim, SSFNet, TOSSIM, TOSSF, TYTHON, EmStar/EmSim/EmTOS, ATEMU, SENSE, SENS, Prowler/JProwler, ModelNet/Nisnet, SwarmNet/Shawn, Glonemo, Avrora	Evaluation in terms of reusability and extensibility, performance and scalability, operating system portability, semantics scripting languages, realism level of virtual environment, graphics, and debug and trace

[39]	008	NS-2, GloMoSim, OPNET, SensorSim, J-Sim, SENSE, OMNeT++, Sidh, SENS, TOSSIM, ATEMU	Overview
[66]	008	J-Sim, OMNeT++, NS-2, OPNET	Comprehensive overview, features, and comparison
[67]	008	J-Sim, OMNeT++, NS-2, ShoX	Overview, installation, implementation and documentation, and visualization and statistics
[68]	008	WISENES, SensorSim, sQualNet, NRL simulator, SWAN, SENSIM, EYES, J-Sim, VisualSense, Prowler, H-MAS, SENSE, TOSSIM, ATEMU, SENS, TOSSF, Em* EmSim, SNAP	Comparison table
[69]	009	NS-2, SensorSim, J-Sim, SENS, TOSSIM, ATEMU, Avrora, EmStar, COOJA	Overview and comparison
[70]	009	NS-2, GloMoSim, J-Sim, OMNeT++, OPNET, QualNet	Comparison table
[71]	009	NS-2, OMNeT++, Prowler, TOSSIM, OPNET	Overview and comparison table
[13]	009	NS-2, Castalia, TOSSIM, COOJA/MSPSim	Overview and comparison table
[33]	010	NS-2, OMNeT++, NesCT, PAWiS, GloMoSim, OPNET, SENSE, Ptolemy II, J-Sim, Cell-DEVS, GTNetS, SystemC, NCTUns2.0, JiST/SWANS, SSFNet	Overview
[72]	010	GloMoSim/QualNet, OPNET, TOSSIM, OMNeT++ (Mobility Framework, MiXiM, Castalia, INET Framework, NesCT), NS-2 (SensorSim), Avrora, J-Sim, ATEMU, EmStar, SENS, SENSE, Shawn	Overview and comparison table
[20]	010	SensorSim, Nsrlsensorsim, Castalia, VisualSense, Viptos, Sidh, Prowler/JProwler, SENS, TOSSIM, ATEMU, Avrora, SENSE, EmStar	Overview
[60]	011	NS-2, OMNeT++, GloMoSim, OPNET, SENSE, TOSSIM, GTSNetS	Overview, comparison table, and performance analysis (CPU time and network lifetime)
[73]	011	NS-2, SensorSim, NRL, OMNeT++, SenSim, Castalia, MixiM, PAWiS, J-Sim, SENSE	Overview
[74]	011	NS-2, OMNeT++ (MiXiM), Worldsens (WSim and WSNet), TOSSIM, COOJA, OPNET, J-Sim, , TRMSim-WSN, WSNim	Comprehensive overview and comparison table

[75]	011	NS-2, TOSSIM, GloMoSim, UWSim, Avrora, SENS, COOJA, Castalia, Shawn, EmStar, SENSE, VisualSense, JProwler	Overview, comparison table, and performance analysis (CPU time and memory usage)
[61]	011	Castalia, MiXiM, TOSSIM, WSNet	Focusing on topology, antenna, radio propagation, noise, radio, medium access control and energy consumption modeling
[56]	011	*Simulators*: Network Simulator (NS-2 and NS-3), Mannasim, TOSSIM, TOSSF, PowerTOSSIM Z, ATEMU, COOJA, GloMoSim, QualNet, SENSE, VisualSense, AlgoSenSim, GTNetS, OMNeT++, Castalia, J-Sim, JiST/SWANS, JiST/SWANS++, Avrora, Sidh, Prowler, JProwler, LecsSim, OPNET, SENS, EmStar, EmTOS, SenQ, SIDnet-SWANS, SensorSim, Shawn, SSFNet, Atarraya, NetTopo, WiseNet, SimGate, SimSync, SNetSim, SensorMaker, TRMSim-WSN, PAWiS, OLIMPO, DiSenS, WISDOM, Sinalgo, Sensoria, Capricorn, H-MAS, Starsim, Motesim, SNSim, SNIPER-WSNim, SNAP, SimPy, Mule, CaVi, Ptolemy, Maple, WISENES, WSNet-Worldsens, WSim, LSU Sensor Simulator, WSNGE, TikTak. *Emulators:* VMNET, ATEMU, EmStar, TOSSIM, AvroraZ/Avrora, Freemote, EmPro, NetTopo, OCTAVEX, SENSE, UbiSec&Sens, Emuli, MSPSim, MEADOWS	Short description
[76]	012	ATEMU, Avrora, Castalia, JProwler, SENSE	Short description
[77]	012	GloMoSim/QualNet, OMNeT++, NS-2, OPNET, J-Sim	Overview and comparison
[32]	012	NS-2, OMNeT++, J-Sim, GloMoSim, SSFNeT, EmStar/EmSim/EmTOS	Overview
[78]	012	SensorSim, TOSSIM, TOSSF, GloMoSim, QualNet, OPNET, EmStar, SENS, J-Sim, Dingo, NS-3, Shawn	Overview and comparison table
[79]	012	NS-2, GloMoSim, J-Sim, OMNeT++, JiST/SWANS, NS-3, SENS, Prowler, TOSSIM, ATEMU, Sidh, OPNET, EmStar	Properties and limitations
[80]	012	NS-2, NS-3, PowerTOSSIM, PowerTOSSIM Z, OMNeT++, MATSNL	Features and comparison table
[7]	012	NS-2, TOSSIM, OMNeT++, J-Sim, ATEMU, Avrora, OPNET, Castalia	Overview, merits and limitations, and comparison table
[57]	012	ATEMU, Avrora, EmSim, Freemote Emulator, MSPSim, TOSSIM, VMNet, WSim, Atarraya, Prowler, Wireless Sensor Network Localization Simulator, WSNet, AlgoSenSim, NetTopo, SENSE, Sensor Security Simulator (S3), Shawn, SIDnet-SWANS, Sinalgo, TRMSim-WSN, Wireless Sensor Network Simulator, WSNimPy, COOJA, J-Sim and Sensor Network Package, SENS, WSN-Sim, NS-2, Mannasim, NRL SensorSim, RTNS, OMNeT++, Castalia, MiXiM, NesCT, PAWiS, SENSIM (SensorSimulator), Ptolemy II, Viptos, VisualSense	Overview and GUI

[81]	012	GloMoSim/QualNet, OPNET, TOSSIM, OMNeT++, MiXiM, Castalia, INET Framework, NesCT, NS-2, Avrora, J-Sim, ATEMU, EmStar, SENS, SENSE, Shawn, MATLAB/SIMULINK Software	Overview and comparison table, performance analysis of USN in MATLAB
[82]	013	NS-2, TOSSIM, OMNeT++, J-Sim, ATEMU, Avrora, SENSE, SensorSim	Overview
[83]	013	NS-2, NS-3, OMNeT++, TOSSIM and its derivatives, Avrora, Worldsens, WISENES, IDEA1	Overview
[84]	013	NS-2, NS-3, TOSSIM, J-Sim, Castalia, QualNet	Overview, merits and demerits
[19]	013	NS-2, J-Sim, OPNET, OMNeT++, GloMoSim, Ptolemy II, JiST/SWANS, NCTUns2.0, SSFNet, TrueTime toolbox (MATLAB), TOSSIM, PowerTOSSIM, TOSSEF&TYTHON, EmStar/EmSim/EmCee, ATEMU, Avrora, Prowler/JProwler, UWSim, Shawn, COOJA/MSPSim	Comparison table and simulator analysis
[85]	013	NS-2, NS-3, OMNeT++, J-Sim	Overview, architecture, advantages and limitations, and comparison table
[86]	013	NS-3, OPNET, GloMoSim, MiXiM, Castalia, J-Sim, Avrora	Overview
[87]	013	NS-2, TOSSIM, GloMoSim, UWSim, J-Sim, SENS, COOJA, SENSE, VisualSense, JProwler, Shawn, Castalia	Comparison table
[88]	013	NS-2, TOSSIM, GloMoSim, QualNet, OPNET, J-Sim, OMNeT++	Overview, architecture, merits and limitations, and comparison table
[89]	013	NS-2, NS-3, OMNeT++, Wireshark (Ethereal), OPNET, GloMoSim/QualNet, J-Sim, GNS3	Comparison table
[90]	013	J-Sim, OMNeT++, NS-2, OPNET	Overview, GUI, comparison table of simulation features
[91]	014	NS-2, NS-3, GNS3, Wireshark (Ethereal), OPNET, OMNeT++, GloMoSim/QualNet, J-Sim, JiST/SWANS, VisualSense, Ptolemy II, TOSSIM, Castalia, EmStar, ATEMU, SENSE, SENS, JProwler, Avrora, COOJA, Shawn	Overview, comparison tables, advantages and disadvantages
[92]	014	NS-2, OMNeT++, J-Sim, OPNET, TOSSIM	Comprehensive overview, features, components, comparison tables, and shortcomings
[93]	014	NS-2, EmStar, GloMoSim, Shawn, UWSim, VisualSense, J-Sim, OMNeT++, Aqua-Sim, QualNet	Underwater USN overview, merits and demerits, and comparison table

[58]	014	*Simulators*: Network Simulator (NS-2 and NS-3), Mannasim, TOSSIM, TOSSF, PowerTOSSIM Z, ATEMU, COOJA, GloMoSim, QualNet, SENSE, VisualSense, AlgoSenSim, GTNetS, OMNeT++, Castalia, J-Sim, JiST/SWANS, JiST/SWANS++, Avrora, Sidh, Prowler, JProwler, LecsSim, OPNET, SENS, EmStar, EmTOS, SenQ, H-MAS, SensorSim, Shawn, NetTopo, Atarraya, SSFNet, WiseNet, SimGate, SimSync, SNetSim, SensorMaker, TRMSim-WSN, PAWiS, OLIMPO, DiSenS, WISDOM, Sinalgo, Sensoria, Capricorn, SIDnet-SWANS, Starsim, SNSim, SNIPER-WSNim, SNAP, SimPy, Mule, CaVi, Ptolemy, Maple, WISENES, WSNet-Worldsens and WSim, LSU Sensor Simulator, WSNGE, TikTak, Motesim, Boris, SmartSim, WSNim, EnergySim, MOB-YOSSIM, AEON, Sensor Security Simulator (S3), Wireless Sensor Network Localization Simulator, Xen WSN Simulator, UWSim, Network in a box (NAB)	Overview, categorization of USN-specific simulators
[94]	015	QualNet, NS-2, NS-3, OPNET modeler, Net Sim, OMNeT++, J-Sim	Overview
[62]	015	NS-2, NS-3, OMNeT++, J-Sim, Mannasim, SensorSim, NRL SensorSim, NCTUns, SSFNet, GloMoSim, QualNet, sQualNet, OPNET, SENSE, DRMSim, NetSim, UWSim, VisualSense, Viptos, Ptolemy II, SENS, Shawn, SIDnet-SWANS, SIDH, NetTopo, WSim/Worldsens/WSNet, WSN Localization Simulator, Prowler, MATLAB, PiccSIM, LabVIEW	Overview, architecture, interface/GUI, and comparison table
[95]	016	NS-2, NS-3, Castalia, MiXiM, PAWiS, WSNet, DANSE, NetTopo, PASES, Sense, TOSSIM, Avrora, COOJA/MSPSim, VIPTOS	Overview, categorization of simulators, comparative study
[96]	016	NS-2, NS-3, GloMoSim, OPNET, OMNeT++, TOSSIM, ATEMU, Avrora, EmStar, SensorSim, NRL SensorSim, J-Sim, Prowler/JProwler, SENS, SENSE, Shawn, SenSim, PAWiS, MSPsim, Castalia, MiXiM, NesCT, SUNSHINE	Overview, component, structure

Overview of USN simulators and emulators

Numerous tools have been developed for simulating and emulating USN. They vary in terms of architecture, features and characteristics, modeling methodology, and performance [91]. Utilizing or developing a simulator/emulator necessitates becoming familiar with the available tools, evaluating their pros and cons, and choosing the appropriate one for the application. This section, therefore, introduces 130 USN simulation and emulation environments and frameworks, as well as their derivatives, which were *originally designed* and *adapted* for USN. The USN simulators and emulators provided in the following are the ones that we found in the literature at the time of this writing. The literature review for tool selection was based on the online publications (i.e., original and survey articles, book chapters, conference proceedings, thesis, and technical reports) as well as the tool developers' websites and tutorials. Specifically, these tools are divided into simulators and emulators. Then, simulators are classified into general-purpose and specific-purpose classes (see Figure 6). General-purpose tools are those that already existed before the emergence the USN concept. The functionality of the simulator/emulator was then extended and adapted for USN purpose. In contrast, specific-purpose class tools are the new simulating/emulating tools that have been created solely for USN purpose. Of these 130 simulators and emulators, the 22 that have been widely used, regularly updated, and well supported by their developers are compared based on multifarious criteria.

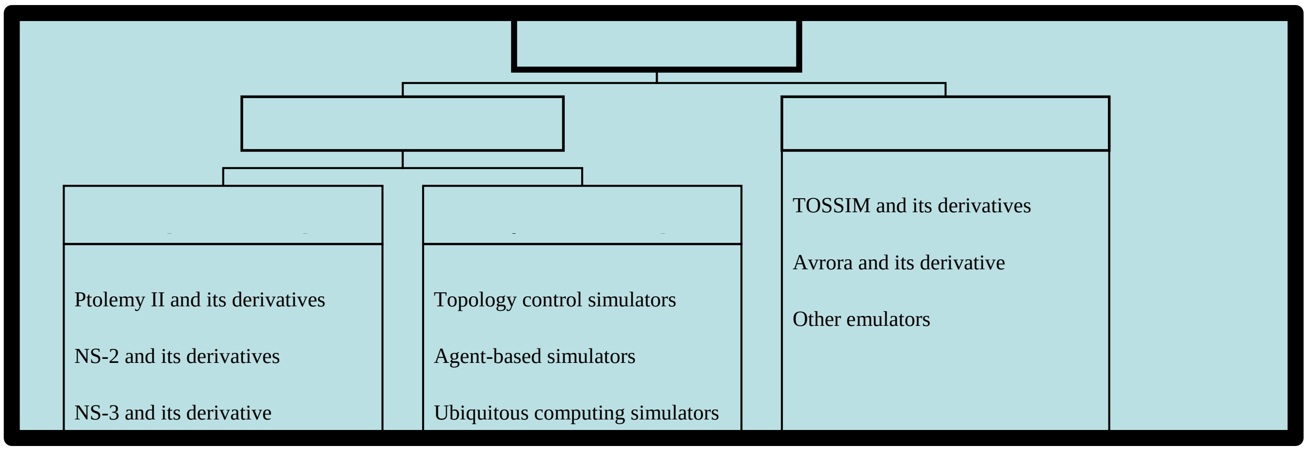
TOSSIM and its derivatives
Avrora and its derivative
Other emulators
Ptolemy II and its derivatives
NS-2 and its derivatives
NS-3 and its derivative
Topology control simulators
Agent-based simulators
Ubiquitous computing simulators

Figure 6. Classification of simulation and emulation environments/frameworks.

USN simulation environments

This sub-section concisely introduces 41 general-purpose USN simulators and their derivatives as well as 66 specific-purpose ones. Meanwhile, the corresponding designer/developer, the latest version, and the software link (if any) are provided for general- and specific-purpose simulation tools in Tables 5 and 6, respectively.

General-purpose simulators

Ptolemy II and its derivatives

1. Ptolemy II is a set of Java packages which support various models of simulation, including continues time, dataflow, and discrete-event. They are an actor-oriented and component-based design of J-Sim (described later as tool no. 33) [97].

2. Viptos (Visual interface between Ptolemy and TinyOS) is a graphical development and simulation environment for TinyOS-based (a component-based, event-driven runtime environment) USN. It is built on Ptolemy II and TOSSIM- an interrupt-level discrete-event simulator for homogeneous TinyOS networks [98, 99].

3. VisualSense is a visual model for ubiquitous and sensor network systems. It was built on Ptolemy II, and preserves the semantics of discrete event, although explicit wires are not required because of changing the mechanism of connecting components [100, 101].

NS-2 and its derivatives

4. NS-2 (Network Simulator-2) is a flexible tool which enables the performance of various protocols to be investigated in different configurations and topologies. In this environment a network of sensors can be built so that protocols and characteristics are available in the real world [102, 103].

5. Mannasim is a module for NS-2 that aims at USN simulation. It provides a sensing model, several application models, several sensor network specific protocols such as LEACH routing protocol, and directed diffusion [104].

6. NRL Sensorsim is an extension of NS-2 and facilitates sensor network simulation. It provides the opportunity to simulate and detect parameters of carbon monoxide concentration, seismic activity, or audible sound. Wireless sensors, phenomena sources, and gateways can constitute the network [57, 87, 105].

7. RTNS (Real Time Network Simulator) software simulates mechanisms of OSs for applications in distributed networks. It combines NS-2 environment and Real Time Operating System SIMulator (RTSim) to simulate CPU in real time [106-108].

8. TRAILS (Toolkit for Realism and Adaptively In Large-scale Simulation) is an extension of NS-2. TRAILS adds functionalities into NS-2 and optimizes its operation. It facilitates the execution of advanced mobility patterns, obstacle presence and disaster scenarios, and failures injection that can dynamically alter over the simulation execution [109].

9. PiccSIM (Platform for integrated communications and control design, Simulation, Implementation and Modeling) is a simulation platform for (wireless) networked control systems that uses NS-2 and MATLAB/SIMULINK tools. It aims to deliver a complete toolset for the design, simulation and implementation of wireless control systems [110].

NS-3 and its derivative

10. NS-3 is a discrete-event network simulator for internet systems targeted primarily for research and educational use. Although the NS-2 simulator is popular, the need for performing core refactoring, integration, software and documentation maintenance, and simulator extension necessitated a new simulator, NS-3. In general, NS-3 is introduced to solve problems present in NS-2. Indeed, NS-3 is not backward compatible with NS-2; it is built to replace NS-2. Specifically, the main contributions that NS-3 can offer comparing with NS-2 are as follows. In NS-2, bi-language system make debugging complex (C++/Tcl), but for NS-3 only knowledge of C++ is enough (single-language architecture is more robust in the long term). NS-3 provides a lower base level of abstraction compared with NS-2, allowing it to align better with how real systems are put together. Some limitations found in NS-2 (such as supporting multiple types of interfaces on nodes correctly) have been remedied in NS-3. NS-3 provides features not available in NS-2, such as an implementation code execution environment (allowing users to run real implementation code in the simulator). NS-3 has better scalability than NS-2. NS-3 has an emulation mode, which allows for the integration with real networks. In contrast, NS-2 is preferred by several users in the community due to the following reasons. Since NS-3 is under development, there is very limited number of models and contributed codes in NS-3 in comparison with NS-2; NS-3 still requires strong community participation to improve it. NS-3 is a new simulator that does not support the NS-2 APIs. Owing to NS-2's long history, it has a more diverse set of contributed modules than does NS-3. However, NS-3 has more detailed models in several popular areas of research (including sophisticated LTE and WiFi models). Picking NS-2 or NS-3 relates to the availability of models and the familiarity of users with the tools; however, the tool that is being actively developed has the priority (i.e., NS-3) [111, 112].

11. Symphony is a simulation framework in association with NS-3, by which the entire processes of actual hardware and software can be modeled. It allows real code adjustment as well as hardware components performance evaluation on applications and protocols in large-scale USN systems [113, 114].

OMNeT++ and its derivatives

12. OMNeT++ (Objective Modular Network Testbed in C++) is a component-based and modular simulation which is designed to simulate communication networks and other distributed systems [115, 116]. It attempts to fill the gap between open-source software (e.g., NS-2) and expensive commercial alternatives (e.g., OPNET) [117].

13. SENSIM (SensorSimulator) is a large-scale sensor network simulator for OMNet++ to compute energy consumption. It is based on a parallel discrete-event system. It integrates common sensor network protocols, including MAC, network, and application as well as an adapted architecture for future protocols [82, 118, 119].

14. LSU SensorSimulator is a customizable framework for USN simulation. It tests and investigates robustness, scaling, networking, and phenomenological issues to find efficient algorithms for distributed sensors [120].

15. Castalia is an application- and discrete-level simulator designed on the top of OMNeT++. It is designed to evaluate various platforms because it is highly parametric and can simulate a wide range of platforms [121, 122].

16. SolarCastalia or Solar Energy Harvesting Wireless Sensor Network Simulator is a USN simulator based on Castalia which uses solar energy as the energy source. It provides high energy density, high conversion efficiency, and periodicity [123].

17. MiXiM (mixed simulator) is a cross-level OMNeT++ modeling framework created for mobile and fixed wireless networks. It consists of basic components including, environment, connectivity, reception and collision, protocol library, and experiment. It also supports visualization, monitoring, and debugging in the simulation process [124].

18. NesCT is a translator from the NesC (Network embedded system C language) programming language to C++ classes for OMNeT++. NesC is an event- and component-driven application simulator. In this way all features of OMNeT++ and Mobile Framework (MF) can be used for simulation [125].

19. PAWiS (Power Aware Wireless Sensors) simulator was developed to facilitate the design and simulation of USN. It is based on the OMNeT++ simulator and the idea of decomposing the node into functional blocks which can be hardware or software [126, 127].

GloMoSim and its derivatives

20. GloMoSim (Global Mobile system Simulator) is a library for parallel simulation of large-scale ubiquitous networks in which each library module simulates a particular wireless protocol in the protocol stack [128, 129].

21. QualNet is a discrete-event simulator and a commercial extension of GloMoSim for scalable network technologies [130]. It has been enhanced over during time by the inclusion of satellite, cellular, and new sensor network library [131]. Considering all the QualNet functionalities plus a real-time network emulation interface, EXata is introduced which enables live hardware integration in a seamless manner with the simulated virtual network models, and live applications to run across the virtual environment [132]. sQualNet is a scalable sensor network simulation framework based on QualNet [133].

22. SenQ is a scalable simulation and emulation framework for sensor networks based on QualNet. It is efficient and flexible for different applications and protocols, and can model battery power and clock drift accurately [134].

Worldsens and its derivative

23. Worldsens is an integrated environment for developing USN applications. It can be used for debugging and performance evaluation because of accurate timing. It consists of two simulators: (1) WSim; and (2) WSNet, which may perform independently or in conjunction during application execution [135, 136].

24. WSNet is an event-driven and large-scale USN simulator [56]. It is designed to simulate the environment with a concentration on physical measures and simulates components of the nods and properties of the radio channel [137].

Other general-purpose simulators

25. AlgoSenSim is an algorithm-oriented framework that is used to simulate network-specific algorithms like localization, distributed routing, and flooding. Its main purpose is to facilitate the implementation and quality analysis of new algorithms [138].

26. NetTopo is an algorithm level, large-scale network simulator which mainly focuses on USN data structure, logic, and presentation of the algorithms. It was developed in Java and provides both simulation and visualization functions [38, 139].

27. SENSE (Sensor Network Simulator and Emulator) is a component-oriented general-purpose network and application level simulator. It supports an energy model that is compatible with USN. The most important point about SENSE is its balanced consideration of modeling methodology and simulation efficiency [140, 141].

28. JiST/SWANS (Java in Simulation Time) is a discrete-event simulation system that embeds simulation time into a virtual machine. It is efficient and transparent within a standard language [142]. SWANS (Scalable Wireless Ad hoc Network Simulator) is a scalable wireless network simulator built atop the JiST platform. It was designed because existing network simulation tools are insufficient for current research needs, and its performance serves as a validation of the virtual machine-based approach to simulator construction [143].

29. Sinalgo (Simulator for Network Algorithms) is an algorithm-based simulator that offers a message passing view of the network which captures well the view of the network device. It concentrates on the verification and testing of network algorithms [144].

30. SimPy is a process-oriented discrete-event simulator. It may be utilized for asynchronous networking or to implement multi-agent systems (with both simulated and real communication) [145].

31. MSPSim is an extensible simulator for the MSP430 microcontroller at the instruction-level. It is designed to be used in a larger sensor network as a component to support cross-level simulation [146, 147].

32. COOJA (COntiki Os JAva) is a cross-level simulator and simulates at many levels of the system simultaneously. It is interchangeable and extensible to change all the levels of the system, and combines low- and high-level simulation of sensor node hardware and behavior in a single simulation [148, 149].

33. J-Sim (formerly JavaSim) is component-based software architecture: ACA-the autonomous component architecture-and a compositional network simulation and emulation environment. Components are the basic entities in the ACA which communicate with one another through their ports [150, 151]. G-JSIM is a GUI tool for USN simulations under J-Sim platform [152].

34. NetSim is a network-based environment for modeling and simulating discrete-event applications to simulate Cisco Systems networking hardware and software. NetSim has been widely used for network design validation in sensor deployment [153, 154]. sNetSim can be utilized for analyzing data packet delivery, probability of discarded packet, and other parameters in USN.

35. OPNET (Optimum Network Performance) Modeler or Riverbed Modeler was the first commercial network simulator developed in 1987. It is a discrete-event, object-oriented, and general-purpose network simulator. It is well-known because of its capability to provide accurate modeling of the radio transmission [7, 155]. The educational version of it is called OPNET IT Guru [156].

36. SSFNeT is a number of Java network models built over the Scalable Simulation Framework (SSF). It is difficult to interact with a simulator because of a command-line user interface; it is therefore only suitable for static applications, but does provide the capability of parallel simulation [157].

37. NCTUns (National Chiao Tung University Network Simulator) is an event-driven simulator based on a Linux OS. It enables several simulations of various protocols used in both wired and wireless IP networks. NCTUns provides high simulation speed when traffic load is light and can be turned into an emulator by slowing down and synchronizing the virtual clock with that in real life [158, 159]. The NCTUns 6.0 version supports large-scale microscopic wireless vehicular network (WVN) simulation [160].

38. SystemC is a modeling platform including libraries to support design abstractions for modeling hardware, software, and networks with the same language [161, 162].

39. Wireshark (formerly Etherreal) is a network simulator and analyzer capable of USN modeling and evaluation. It is based on the packet analysis as well as applicable for trouble shooting, examining network security, and the development of software and communication protocols. It supports over 750 protocols, which may be exceeded due to its open-source specification [163, 164].

40. MATLAB SIMULINK can be used as an USN simulator [165]. The unique feature of this simulation tool is its capability to determine the effects of different channel parameters such as signal to noise ratio, attenuation, and interference [81, 166]. The authors of a research contribution [167] have developed an interface between MATLAB and OPNET to perform much stronger simulation.

41. LabVIEW is a development environment suitable for visualizing, creating, and coding engineering systems. It enables USN simulation. By programming sensor nodes, an individual can customize the node behavior to increase acquisition performance, interface directly with sensors, and extend battery life [168].

Table 5. General descriptions of general-purpose USN simulators.

o.	Simulation Environment	Designed By	Latest Version	Released Date	Software Link
	Ptolemy II	University of California, Berkeley, USA	10.0.1	17 December 2014	http://ptolemy.eecs.berkeley.edu/ptolemyII/
	Viptos	University of California, Berkeley, USA	1.0.2	9 February 2007	http://ptolemy.berkeley.edu/viptos/
	VisualSense	University of California, Berkeley, USA	8.0.1	28 October 2010	http://ptolemy.berkeley.edu/visualsense/
	NS-2	Lawrence Berkeley National Laboratory (LBNL), USA	NS-2.35	4 November 2011	http://www.isi.edu/nsnam/ns/
	Mannasim	Federal University of Minas Gerais, Brazil	NS-2.29 + patch 2.29	16 August 2006	http://www.mannasim.dcc.ufmg.br/
	NRL Sensorsim	U.S. Naval Research Laboratory	NS-2.27	6 October 2005	http://www.nrl.navy.mil/itd/ncs/products/sensorsim
	RTNS	Sant'Anna School of Advanced Studies, Italy	1.0	25 August 2008	http://rtns.sssup.it/RTNSWebSite/RTNS.html
	TRAILS	University of Patras, Greece	N/A	2008	N/A
	PiccSIM	Aalto University, Finland	1.16	4 November 2013	http://wsn.aalto.fi/en/tools/piccsim/
0	NS-3	University of Washington and Georgia Institute of Technology, USA	NS-3.26	3 October 2016	https://www.nsnam.org/
1	Symphony	Stanford University, USA and Lulea University of Technology, Sweden	N/A	2015	https://bitbucket.org/Northshoot/symphony/overview

2	OMNeT++	Technical University of Budapest, Hungary	5.1	22 December 2016	https://omnetpp.org/
3	SENSIM	Louisiana State University, USA	3.1	28 October 2005	http://csc.lsu.edu/~iyengar/publications_sensor.html#papers
4	LSU SensorSimulator	Louisiana State University, USA	N/A	2005	N/A
5	Castalia	National ICT, Australia	3.2	30 March 2011	https://castalia.forge.nicta.com.au/index.php/en/
6	SolarCastalia	Soongsil University, Republic of Korea	N/A	N/A	N/A
7	MiXiM	University of Paderborn and Technical University of Berlin, Germany; Delft University of Technology, the Netherlands	2.3	8 March 2013	http://mixim.sourceforge.net
8	NesCT	University of Twente, the Netherlands; Yeditepe University, Turkey	OMNeT 4.2	4 August 2011	http://nesct.sourceforge.net/
9	PAWiS	Technical University of Vienna, Austria	2.0	1 July 2008	http://pawis.sourceforge.net/
0	GloMoSim	University of California, Los Angeles (UCLA), USA	2.03	December 2000	http://pcl.cs.ucla.edu/projects/glomosim/
1	QualNet	Scalable Network Technologies (SNT), Inc., USA	7.2	14 May 2014	http://web.scalable-networks.com/content/QualNet
2	SenQ	University of California, Los Angeles (UCLA), USA	N/A	2007	N/A
3	Worldsens	Senslab, France	N/A	2007	https://www.iot-lab.info/

4	WSNet	INRIA, France	9.07	16 July 2009	http://wsnet.gforge.inria.fr/
5	AlgoSenSim	University of Geneva, Switzerland	0.9.2.2	21 September 2006	http://tcs.unige.ch/doku.php/code/algosensim/overview
6	NetTopo	Osaka University, Japan; National University of Ireland, Ireland; National Ilan University, Taiwan; Seoul National University, Republic of Korea; Simula Research Laboratory, Norway; University of Oslo, Norway	1.3	1 August 2008	http://sr.deri.ie/nettopo/index.htm
7	SENSE	Renseelaer Polytechnic Institute, USA	3.1	19 November 2008	http://www.ita.cs.rpi.edu/
8	JiST/SWANS	Cornell University, USA	1.0.6	March 2005	http://jist.ece.cornell.edu/
9	Sinalgo	ETH Zurich, Switzerland	0.75.3	8 April 2008	http://disco.ethz.ch/projects/sinalgo/
0	SimPy	Team SimPy	3.0.10	12 June 2016	http://simpy.readthedocs.org/en/latest/
1	MSPSim	Swedish Institute of Computer Science, Sweden	0.9X	30 April 2009	http://sourceforge.net/projects/mspsim/
2	COOJA	Swedish Institute of Computer Science, Sweden	2.7	15 November 2013	http://www.contiki-os.org/
3	J-Sim	University of Illinois at Urbana-Champaign, USA	1.3 + patch4	5 July 2006	https://sites.google.com/site/jsimofficial/
4	NetSim	Tetcos in association with Indian Institute of Science, India	8.3	N/A	http://tetcos.com/netsim_gen.html
5	OPNET	Massachusetts Institute of Technology (MIT), USA	18.5.1	28 April 2016	http://www.riverbed.com/products/performance-management-control/opnet.html

6	SSFNeT	SSF Research Network	2. 0.0	15 January 2004	http://www.ssfnet.org/homePage.html
7	NCTUns	National Quemoy University, Taiwan	6. 0	201 0	http://nsl.cs.nctu.edu.tw/NSL/nctuns.html
8	SystemC	University of Verona and Polytechnic University of Turin, Italy	2. 3.1	23 April 2014	https://github.com/systemc/systemc-2.3
9	Wireshark (Ethereal)	Wireshark Team	2. 2.3	14 December 2016	https://www.wireshark.org/
0	MATLAB SIMULINK	Mosul University, Iraq	N/ A	201 1	http://www.mathworks.com/
1	LabVIEW	National Instruments Corporation, Germany	20 15	201 5	http://www.ni.com/labview/

Specific-purpose simulators

Topology control simulators

42. Atarraya is a discrete-event simulation tool to test the implementation of topology control protocols in USN. This simulator encompasses structures for designing topology construction and maintenance protocols [169, 170].

43. Cell-DEVS (Cell-Discrete-Event systems Specifications) is a discrete-event simulator that is used to model systems that can be represented as cell spaces. It is an efficient simulation model to implement topology control algorithms for a large-scale USN [171, 172].

Agent-based simulators

44. ABMQ (Agent-Based Modeling and Simulation) is a platform based on Qt Application Framework, appropriate for modeling and simulation of self-organization in wireless networks, and particularly Mobile Ad Hoc Networks (MANETs) [173].

45. MASON (Multi-Agent Simulator Of Neighborhoods/Networks) is a rapid discrete-event multi-agent simulation library written in Java. It is comprised of a model library, and 2D and 3D visualization tools [174-176].

46. RepastSNS (Recursive Porous Agent Simulation Toolkit Sensor Network Simulation) is an event-based simulator developed for testing sensor networks from a multi-agent perspective. This platform is an extension of Repast3 [177] as it has additional features for sensor network simulation [178, 179]. RepastSNS has two advantages: (1) it provides many abstraction level descriptions, and (2) it is easy to insert USN components for simulation [180].

47. NetLOGO is a free platform for multi-agent programming and modeling. It provides the opportunity to simulate USN projects, e.g., energy efficiency [181] and data dissemination flooding technique [182].

48. SXCS (SensomaX Companion Simulator) is a hybrid agent-based multi-operational simulator aimed at the simulation of multiple concurrent applications in USN. It is designed for emulating Sensomax [183] middleware, which

is an agent-based middleware with multiple concurrent application support for dynamic data gathering in large-scale USN [184].

Ubiquitous computing simulators

49. UbiWise is a simulator for ubiquitous computing. It focuses on the way that devices compute communications through the physical environment. Ubiwise not only simulates the prototype of devices and protocols in the network, but also simulates the physical environment [185, 186].

50. UbikSim is a simulator for ubiquitous computing that aims to reduce the features of services experiments and applications for treatment appertain related to both the physical environment and users. It proposes substantial techniques for implementing new sensor configurations, e.g., type of event detection or the required range of coverage [187, 188].

51. TATUS is a ubiquitous computing simulator that enables researchers to define and test various scenarios. Meanwhile, a part of software-under-test may be connected to the simulator in order to develop its own representation of the world [189].

Underwater simulators

52. UWSim is a simulator designed for Underwater USN (UUSN). This simulator considers factors that affect USN underwater and adapts scenarios with this condition, such as providing low bandwidth, low frequency, high transmission power, and limited memory [190, 191].

53. SUNSET is an environment for simulation, emulation and also testing (underwater) various communication protocols. Its functionality is at MAC and Routing Protocols based on NS-2 and the extension NS-2-Miracle [192]. By using SUNSET, different acoustic modems and sensing devices can be implemented [193-195].

54. SUNRISE is another UUSN, designed for sensing, monitoring and actuating underwater surroundings. It performs over SUNSET platform for designing, implementing, and validating USN protocols [196, 197].

55. DESERT (DEsign, Simulate, Emulate and Realize Testbeds for underwater network protocols) is a complete set of public C/C++ libraries which support the application and transport layers through the network, data link, and physical layers [198, 199].

56. RECORDS (Remote Control Framework for Underwater Networks) is an open-source environment for underwater networks composed of acoustic nodes. MAC, network, transport, and application layers all are supported by the RECORDS [200, 201].

57. Aqua-Net is a generic architecture for underwater sensor networks. Aqua-Net enables a powerful networking solution kit which facilitates UUSN study and application development [202, 203].

58. SeaLinx is multi-instance protocol stack architecture for underwater acoustic networking. It is a Linux implementation of Aqua-Net and enables users to exploit their hardware more efficiently by allowing applications to run simultaneously on a modem while also providing better support for cross-layer communication [204, 205].

59. Aqua-Net Mate is a real-time virtual channel modem simulator for Aqua-Net that supports underwater networks communication [206].

60. Aqua-Lab is an underwater acoustic sensor network lab testbed for UUSN. It is comprised of hardware, software, program library, an emulator, and real acoustic communication channels [207, 208].

61. Aqua-Sim is another underwater sensor simulator based on NS-2 which can efficiently simulate the acoustic signal attenuation and packet collisions within water. It is extensible, flexible, and independent of the wireless packages [209, 210].

62. Aqua-Tune provides a standardized platform for testing and bridging the gap between modeling, simulation, and real world field experience for Underwater Networks [211].

63. Aqua-GloMo is an acoustic-based communication simulator built on GloMoSim simulator. It is designed for network layers protocols and physical layer protocols of UUSN [212].

64. Aquatools is a simulation toolkit targeted for simulating underwater acoustic networks with static and mobile nodes. It works in physical layer, MAC layer, routing layer, and energy consumptions schemas [213].

65. UANT (Underwater Acoustic Networking plaTform) aims to address the constantly changing underwater acoustic channel with re-configurability. It supports the physical and MAC layer of the ISO Model [214, 215].

66. WOSS (World Ocean Simulation System) is a simulator based on NS-2 for underwater networks which incorporates a ray tracing tool for a more realistic modeling of underwater propagation [216, 217].

67. AUWCN (Acoustic Underwater Channel and Network) simulator aims to alleviate the inappropriate simplifications and to reproduce most effects existing in the physical acoustic underwater channel [218].

68. SAMON (Ocean Sampling Mobile Network) is a simulator testbed designed to enable simulation of ocean sampling missions involving multiple heterogeneous unmanned underwater vehicles prior to in-water experimentation [219].

69. UsNeT (Underwater Sensor Network Simulation Tool) is developed for underwater communications. It allows real-time process-based simulation and enables three-dimensional deployment [220].

Other specific-purpose simulators

70. Prowler/JProwler (Probabilistic Wireless Sensor Network Simulator) is an event-driven ubiquitous network simulator which is designed to simulate MICA motes running TinyOS in addition to generic ubiquitous networks. It is used for application testing (deterministic mode), wireless communication channel, and low-level node protocol simulations (probabilistic mode). It supports different plug-ins and any number of sensor nodes in a dynamically changing environment. JProwler is a java-based prowler [20, 43, 79, 221].

71. Wireless Sensor Network Localization Simulator is a simple, scalable, and discrete-event simulation system. It engages several mobility models including, Random Waypoint, Modified Random Waypoint, Random Direction, Modified Boundless, Manhattan, Freeway, and RPGM [222].

72. Sensor Security Simulator (S3) is a research simulator for evaluating security problems in large-scale sensor networks. It supports the analysis of the impact of selected nodes and encryption keys that compromising the network operation and security [223].

73. Shawn is a discrete-event customizable simulator for USN. It is designed to simulate hundreds of sensors in network [224, 225]. It simulates only the caused effects rather than simulating the effects of a phenomenon which provides a performance increase [72].

74. SIDnet-SWANS (Simulator and Integrated Development Platform for Sensor Networks Applications) is a simulation-based environment that aims to observe the behavior of algorithm protocols in conditions like phenomena fluctuations or sudden loss of service [56]. It associates with a graphical representation of the network, and supports defining various phenomenon such as temperature, humidity, and object movements [226, 227].

75. TRMSim-WSN (Trust and Reputation Models Simulator for Wireless Sensor Networks) is designed to study and compare trust and reputation models over USN and compare the result against other models [228, 229].

76. WSNimPy is based on the discrete-event simulator SimPy. It has developed in two versions: (1) WSNimPy (trace), and (2) WSNimPy (synthetic). The first uses trace data from the WSN Profiler, and the second uses a synthetic radio model to simulate communications [230].

77. SENS (Sensor, Environment and Network Simulator) is a sensor network simulator for USN applications that is flexible and extensible to change components for applications, network communication, and the physical environment. It

supports the development of dependable applications by using diagnostic facilities such as power utilization analysis [231, 232].

78. IFAS (Interactive Flexible Ad hoc Simulator) is a modern and novel approach of ad-hoc simulators. This simulator efficiently accelerates the design and debugging-process of new algorithms by providing unique viewing, debugging, tuning, and interactive capabilities [233].

79. Sidh is an efficient and large-scale USN simulator in which networks with thousands of nodes in real-time can be supported. It is component-based and flexible in the face of different environmental conditions, sensors, and simulation detail [41].

80. SenSor is an algorithmic simulator that works at a high abstraction level in USN domain. It supports a graphical user interface and several extended classes by the user to run simulations [234].

81. Dingo is a fork of the SenSor project. Actually, Dingo is a workbench for prototyping algorithms in USN which uses a top-down methodology for design. Since it is not limited to a particular platform, full functionality of a programming language is usable. It has a fixed API, a simple GUI, and base classes which are extensible by the user [235].

82. SNAP (Sensor Network Asynchronous Processor) is an integrated hardware simulation-and-deployment platform for USN. It lets parallel network simulation with a particular synchronization protocol that is called Time Based Synchronization (TBS). It can be used to build physical as well as simulation nodes [236, 237].

83. GTSNetS is an extension of the Georgia Tech Network Simulator (GTNetS) suitable for USN. It is a large-scale network simulator capable of handling several hundred thousand nodes. It is flexible and capable of implementing diverse protocols, applications, and sensors as well as different energy and accuracy models [238, 239].

84. IDEA1 (hIerarchical DEsign plAtform for sensor networks exploration) is a component-based simulation framework for USN. It is based on SystemC and C++ language. IDEA1 supports transaction-level modeling, and enables easy design space exploration availability [83, 240, 241].

85. WiseNet is a simulation environment aimed at design, application, and evaluation of secure routing protocols for USN. WiseNet enables the systematic development and investigation of security features of secure and intrusion-tolerant routing protocols [242].

86. SimGate is a sensor network simulator which simulates the Intel Stargate device, an element used as a processing, storage, and network gateway unit in sensor network. It is capable of capturing components, including processor, communications, and peripherals [243].

87. SimSync is mainly a time synchronization simulator for sensor networks using a Mica2-a testbed to simulate execution of time synchronization algorithm. The structure of the program in this simulator is table-driven and it can model the clock drift efficiently [244].

88. SensorMaker is a USN simulator for scalable and fine-grained instrumentation. It supports several results such as position, residual energy, energy distribution map, and routing probabilities for each node. It is flexible and provides different kinds of instrumentations [245].

89. OLIMPO is a discrete-event simulation tool designed to ease the design, development, and testing of communication protocols in a sensor network. It provides users with the capability to change various simulation parameters including timers, radio range, and random process [246].

90. DiSenS (DIstributed SENsor network Simulator) is a scalable distributed simulation system for sensor networks. In addition to emulating many sensors, it includes a distributed memory parallel cluster system and extensible radio models [247]. S^2DB is a debugger for sensor networks based on DiSenS [248].

91. WISDOM (Wsn mIddleware Service moDules simulatiOn platforM) is designed to simulate the system architecture and components of middleware in USN. It is capable for adding and testing various middleware algorithms [249].

92. Sensoria is a large-scale, user-friendly, and component-based simulator that can efficiently adapt to different levels of details and accuracy in simulation. Its GUI enables users to implement various scenarios and supports different graphical output formats for the results [250].

93. Capricorn is a large-scale and discrete-event simulator for USN [251].

94. H-MAS (Heterogeneous, Mobile, Ad-hoc Sensor network) is mainly designed to provide a convenient platform for evaluating MAS configurations in various network layers. It also provides a user-friendly visualization tool for non-expert users [252].

95. SnSim is an event-driven software tool that is used to balance data quality and USN lifetime in sensor networks. It includes power consumption elements as well as a graphical interface to investigate various aspects of development [253].

96. SNIPER-WSNim is a simulator specifically designed for USN. It aims to analyze the nodes behavior in USN as well as studying routing protocols and clustering [254].

97. CaVi is a simulation environment to model a network as a collection of nodes in a sensor network. It provides a uniform interface to simulator Castalia for Mont-Carlo simulation and model checking, as well as tools to evaluate statistical information from simulation [255].

98. WISENES (WIreless SEnsor NEtwork Simulator) is a packet-level simulator, designed to simulate high-level USN protocols. It also provides accurate information about their performance in a real environment [68].

99. WSNGE is a scalable and user-friendly simulator with an extensible environment for USN. All functions can be run in scripting and visually, and users can view the results in a graphical environment [256].

100. TikTak is a scalable simulator for USN. The emulation at the protocol-level increases the simulation speed, and low-level hardware emulation provides the ability to simulate the program and stack latency. It also, allows testing and debugging embedded codes [257].

101. ShoX (Scalable ad HOC networK Simulator) is an object-oriented USN simulator. The architecture of the system is an OSI seven-layer model in which all layers are derived from an abstract super class. The most important advantage of this simulator is its comprehensive GUI for configuration, visualization, and statistics demonstration [67, 258].

102. PASENS (Parallel Sensor Network Simulator) is an optimal-synchronous parallel discrete-event simulator that was designed to decrease the period of simulation in large-scale USN with large amount of details [16, 259].

103. Glonemo (GLObal NEtwork MOdel) is a framework for constructing ad-hoc sensor network models and analyzing them globally at different levels of abstraction. This means it models the hardware including a single node, the protocol layers, the application code, and the abstract model of the physical environment as sensed via sensors [260, 261].

104. Maestro is a tool for orchestrating simulations in clouds. It enables the entire application to be simulated simultaneously using numerous sensors and actuator devices in an USN and the functionality of the whole system to be evaluated. Maestro can also be incorporated in parallel multiple serial simulations [262].

105. CupCarbon is a multi-agent and discrete USN design and simulation platform. The sensors can be deployed in Open Street Maps (OSM) and evaluate the behavior of the network and its cost [263, 264].

106. TimSim is a time-step-based wireless ad-hoc network simulator. It directly simulates the source code of wireless ad-hoc network application and is able to represent the transmitted data at the bit level [265].

107. JSensor provides parallel simulation for USN and distributed systems. It enables synchronous and asynchronous simulation of large sensor networks. The kernel of JSensor is based on Sinalgo. The multi-core architecture allows hundreds of nodes to be simulated simultaneously [266, 267].

Table 6. General description of specific-purpose simulators.

o.	Simulation Environment	Designed By	La test Version	Rel eased Date	Software Link
2	Atarraya	University of North, Colombia; University of South Florida, USA	1. 3 beta	Sept ember 2009	http://www.cse.usf.edu/~labrador/Atarraya/
3	Cell-DEVS	University of Ottawa and Carleton University, Canada	N/ A	200 9	http://cell-devs.sce.carleton.ca/mediawiki/index.php/Main_Page
4	ABMQ	Sharif University of Technology, Iran; University of Oulu, Finland	N/ A	201 3	N/A
5	MASON	George Mason University, USA	19	19 September 2015	http://cs.gmu.edu/~eclab/projects/mason/
6	RepastSNS	University of Barcelona, Spain	N/ A	201 3	http://www.iiia.csic.es/~mpujol/RepastSNS/
7	NetLOGO	Northwestern University, USA	6	Dec ember 2016	http://ccl.northwestern.edu/netlogo/index.shtml
8	SXCS	University of Bristol, UK	N/ A	201 3	N/A
9	UbiWise	HP Laboratories Palo Alto, USA	N/ A	200 3	http://home.comcast.net/~johnjbarton/ubicomp/ur/ubiwise/
0	UbikSim	University of Murcia, Spain	2. 0	201 1	https://github.com/emilioserra/UbikSim/wiki
1	TATUS	University of Dublin, Ireland	N/ A	200 4	N/A

2	UWSim	University of Delhi, Indian Institute of Technology, India; Monmouth University, USA	N/A	2008	http://www.irs.uji.es/uwsim/
3	SUNSET	Sapienza University of Roma, Italy	2.0	2013	http://reti.dsi.uniroma1.it/UWSN_Group/index.php?page=sunset&sec=tech_desc
4	SUNRISE	Sapienza University of Roma, Italy	N/A	N/A	http://fp7-sunrise.eu/
5	DESERT	University of Padova, Italy	2.1.0	2015	http://nautilus.dei.unipd.it/desert-underwater
6	RECORDS	University of Padova, Italy	N/A	N/A	http://nautilus.dei.unipd.it/desert-underwater
7	Aqua-Net	University of Connecticut, USA	N/A	2009	http://obinet.engr.uconn.edu/wiki/index.php?title=Aqua-Net&redirect=no
8	SeaLinx	University of Connecticut, USA	N/A	2013	http://www.oceantune.org/index.php/79-ocean-tune/network-solution/71-sealinx
9	Aqua-Net Mate	University of Connecticut, USA	N/A	2013	N/A
0	Aqua-Lab	University of Connecticut, USA	N/A	2007	http://obinet.engr.uconn.edu/wiki/index.php?title=Aqua-Lab&redirect=no
1	Aqua-Sim	University of Connecticut, USA	Beta	2009	http://obinet.engr.uconn.edu/wiki/index.php?title=Aqua-Sim&redirect=no
2	Aqua-Tune	University of Connecticut, USA	N/A	2009	N/A
3	Aqua-GloMo	University of Delhi, India; Monmouth University, USA	N/A	2012	N/A
4	Aquatools	Jacobs University Bremen, Germany	N/A	2010	N/A
5	UANT	University of California, Los Angeles (UCLA), USA	N/A	2009	https://github.com/nesl/uant

6	WOSS	University of Padova, Italy	1. 3.5	21 February 2013	http://telecom.dei.unipd.it/ns/woss/
7	AUWCN	University of Applied Sciences Kiel, Germany	N/ A	201 2	N/A
8	SAMON	Pennsylvania State University, USA	N/ A	200 1	N/A
9	UsNeT	University of Portsmouth, UK	N/ A	201 3	N/A
0	Prowler	Vanderbilt University, USA	1. 25	28 January 2004	http://www.isis.vanderbilt.edu/projects/nest/prowler/
1	Wireless Sensor Network Localization Simulator	Al-Azhar University, Egypt	2. 1	Jun e 2013	http://www.codeproject.com/Articles/606364/Wireless-Sensor-Network-Localization-Simulator-v
2	Sensor Security Simulator (S3)	Masaryk University, Czech Republic	2. 0.0.26	200 8	http://www.fi.muni.cz/~xsvenda/s3.html
3	Shawn	Braunschweig University of Technology, Germany	N/ A	8 November 2006	https://github.com/itm/shawn/
4	SIDnet-SWANS	Joint Lab of Samsung Advanced Institute of Technology & The City University of New York, plus Northwestern University, USA	1. 5.6	6 January 2011	http://users.eecs.northwestern.edu/~ocg474/SIDnet.html
5	TRMSim-WSN	University of Murcia, Spain	0. 5	25 March 2012	http://ants.inf.um.es/~felixgm/research/trmsim-wsn/
6	WSNimPy	Colorado School of Mines, USA	1. 0	3 July 2010	N/A
7	SENS	University of Illinois at Urbana-Champaign (UIUC)	3. 1	31 January 2005	http://osl.cs.illinois.edu/sens/

8	IFAS	Haifa University	N/A	2007	N/A
9	Sidh	University of Maryland, USA	N/A	2005	N/A
0	SenSor	University of Wolverhampton, UK	N/A	2006	N/A
1	Dingo	University of Wolverhampton, UK	N/A	2008	https://code.google.com/p/dingo-wsn/
2	SNAP	Cornell University, USA	N/A	2003	http://vlsi.cornell.edu/~rajit/ps/snap.ps.gz
3	GTSNetS	Georgia Tech University, USA	N/A	10 October 2008	http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/index.html
4	IDEA1	University of Lyon, France	2.0	5 January 2015	http://idea1inl.free.fr/IDEA1/
5	WiseNet	Google Project	r329	June 2011	https://code.google.com/p/secWSNim/
6	SimGate	University of California, Santa Barbara, USA	N/A	2006	N/A
7	SimSync	Chinese Academy of Sciences and Hefei University of Technology, China	N/A	2006	N/A
8	SensorMaker	Seoul National University and Soongsil University, Republic of Korea	N/A	2008	N/A
9	OLIMPO	University of Seville, Spain	N/A	2004	N/A
0	DiSenS	University of California, Santa Barbara, USA	N/A	2007	N/A

1	WISDOM	Nanyang Technological University, Singapore	N/A	2008	N/A
2	Sensoria	The Hashemite University, Jordan	N/A	2007	N/A
3	Capricorn	Wayne State University, USA	N/A	2004	N/A
4	H-MAS	University of Notre Dame, USA	N/A	2003	N/A
5	SnSim	Northeastern University, China	N/A	2010	N/A
6	SNIPER-WSNim	University of Technology, Australia; Wroclaw University of Technology, Poland	N/A	2009	N/A
7	CaVi	National ICT Australia and Macquarie University, Australia; Oxford University Computing Laboratory, UK	N/A	2008	N/A
8	WISENES	University of Tampere, Finland	N/A	2008	N/A
9	WSNGE	University of Geneva, Switzerland; University of Patras, Greece	N/A	2009	N/A
00	TikTak	University of Rome, Italy	N/A	2010	N/A
01	ShoX	T. S. developers	1.0	28 January 2008	http://shox.sourceforge.net/
02	PASENS	Yonsei University, Republic of Korea	0.90	2 August 2007	http://user.chol.com/~legnamai/pasens/
03	Glonemo	France Telecom R&D and VERIMAG, France	N/A	2006	http://rml.lri.fr/glonemo/

04	Maestro	Lulea University of Technology, Sweden	N/A	2014	N/A
05	CupCarbon	Lab-STICC, Virtualis, IEMN, XLim, France and UCD Dublin, Ireland and LIMED, Algeria	2.9.1	2016	http://www.cupcarbon.com/
06	TimSim	Shandong University, China	N/A	2013	N/A
07	JSensor	Federal University of Ouro Preto, Brazil	N/A	August 2014	http://www.joubertlima.com.br/JSensor/

USN emulation environments

This sub-section introduces 23 emulation tools and their derivatives. For each tool, the corresponding designer/developer, the latest version, and the software link (if any) are provided in Table 7.

TOSSIM and its derivatives

1. TOSSIM is an efficient and scalable simulator for TinyOS USN. It has a simple operation environment by using a probabilistic bit error model, and supports various network interactions which make it expensive. It can also be used to discover bugs from network bit-level MAC interactions to queue overflows in ad-hoc routing protocol [268, 269]. The GUI of TOSSIM is known as JTOSSIM. Also, mTOSSIM is a simulator that estimates the battery lifetime in USN [270].

2. PowerTOSSIM z is a plug-in which models power consumption for TOSSIM. The PowerTOSSIM [271] plug-in for power consumption has not been fully imported to new versions of TOSSIM, so PowerTOSSIM z is developed to simulate MICAz motes. It offers a non-linear energy model to capture the behavior of modern batteries [272, 273].

3. TOSSF (TinyOS Scalable Simulation Framework) can create application types on the fly for use in the initialization of the model. It also provides the TinyOS programmer with a set of scripts which adapt the source code to run in the simulator. TOSSF was designed to enhance TOSSIM scalability [274].

4. TYTHON is an extension to TinyOS’s TOSSIM simulator scripted in Python. Given its valuable library of scripting, TYTHON empowers users to develop dynamic and reproducible simulation scenarios [275, 276].

5. Mule is a hybrid simulator based on TOSSIM. It is mainly designed to test and debug USN through a combination of debugging multiple simulated motes on a host with message transmission and sensor data acquisition of physical motes [277].

Avrora and its derivative

6. Avrora is a scalable cycle-accurate sensor network simulator with precise timing. It is an instruction-level sensor network simulator which supports more than 10,000 nodes quickly and can handle 25 nodes in real-time [278, 279]. AvroraZ [280] is an extension to Avrora for improving MicaZ support and thereby provides IEEE 802.15.4 compliant emulations.

7. AEON (Accurate Prediction of Power Consumption) is an extension of Avrora that is utilized to quantitatively predict the energy consumption and estimation of sensor networks [281].

Other emulators

8. ATEMU is a fine-grained sensor network simulator aimed at bridging the gap between actual sensor network deployments and sensor network simulations. The main advantage of ATEMU is its ability to simulate a heterogeneous sensor network [282, 283].

9. EmPro is an Environment/Energy Emulation and Profiling Platform for USN. It is designed to emulate environmental conditions, and all inputs to a sensor system including power sources, radio activities, and inputs from external sources. In profiling mode, it can capture the behavior of USN [284].

10. OCTAVEX ubiquitous sensor framework is intended to assist a wide range of end users, including systems integrators, software developers, and original equipment manufacturers in developing and handling USN. This framework is a backbone that provides user with the ability to implement an inexpensive end-to-end solution quickly and easily [56, 285].

11. SensEH is software framework that enables developers to manipulate the power and speed of a simulation and the reality and accuracy of experiments. It relies on COOJA for emulating the actual code [286].

12. HarvWSNet is considered a suitable tool for the simulation of the network protocols and the lifetime of energy harvesting (EH) of USN. HarvWSNet is based on WSNet and MATLAB. Users may build multi-node network scenarios while imposing a concise description for node energy harvesting, management subsystem, and its time-varying environmental parameters [287].

13. UbiSec&Sens is an architecture for medium and large-scale USN. It contains a comprehensive toolbox of security aware ingredients for sensor network application progression [288].

14. Emuli (Emulated Stimuli) is a method of effectively substituting synthetic data for sensor data on physical wireless nodes. It stores the model parameters rather than recording and playing back spot measurements that cause a compact data memory footprint [289].

15. MEADOWS is a software framework for Modeling, Emulation, and Analysis of Data Of Wireless Sensor networks. The framework is capable of answering questions about sensor query processing [290].

16. Freemote Emulator is a lightweight and visual emulator for USN. It provides a lightweight emulation tool to combine motes and predefined code templates, as well as a layered architecture to produce quick running codes for nodes [291, 292].

17. VMNet (Virtual Mote Network) is a sensor network simulator designed to realistically emulate USN at the CPU clock cycle level. It reports the performance of application in response time, as well as power consumption and emulates peripherals in detail [293, 294].

18. WSim is a sensor node platform emulator which is designed to run, analyze, and debug applications for sensor networks [57]. It provides hardware platform simulation by microcontroller binary codes and simulates its behavior, as well as any event that occur in the platform [135, 295].

19. EmStar is software for developing and deploying USN on Linux. It includes libraries to implement message-passing IPC primitives to deploy, simulate, emulate, and visualize live systems. Further, it contains services to support sensing, networking, and time synchronization [296, 297].

20. WiEmu is an open-source agent-based simulator and emulator for heterogeneous USN. Using WiEmu enables the evaluation of the network architecture, topology, and protocols, even when they running on real testbeds [298, 299].

21. WiSeREmulator is an emulation framework for structural health monitoring. It is comprised of two main components: a testbed of wireless sensor nodes, and a software emulation environment [300].

22. SUNSHINE is a scalable hardware-software emulator for sensor network applications. It provides data exchanges and time synchronizations across different simulation domains and simulation accuracy levels [301, 302].

23. CORE (Common Open Research Emulator) is composed of Python modules and GUI for building emulated USN networks. These networks may be connected to live networks [303].

Table 7. General description of USN emulators.

o.	Emulation Environment	Designed By	Latest Version	Released Date	Software Link
	TOSSIM	University of California, Berkley, USA	TinyOS 2.1.2	20 August 2012	http://www.tinyos.net
	PowerTOSSIM z	Trinity College Dublin, Ireland	4.0	26 November 2014	https://www.scss.tcd.ie/disciplines/computer_systems/ccg/software/powertossimz/
	TOSSF	Dartmouth College, USA	N/A	2002	N/A
	TYTHON	University of California, Berkley, USA	N/A	2005	http://www.tinyos.net/tinyos-1.x/doc/tython/tython.html
	Mule	Kent State University, USA	N/A	2004	N/A
	Avrora	University of California, Los Angeles (UCLA), USA	1.7.117	21 August 2013	http://compilers.cs.ucla.edu/avrora/
	AEON	University of Tubingen, Germany	N/A	2005	N/A
	ATEMU	Maryland University, USA	0.4	31 March 2004	http://www.cshcn.umd.edu/research/atemu/
	EmPro	University of California, Irvine, USA	N/A	2006	N/A
0	OCTAVEX	Octave Technology, Inc.	Beta	2005	https://www.millennium.berkeley.edu/pipermail/tinyos-help/2005-September/012224.html
1	SensEH	University of Trento, Italy	N/A	2014	N/A

2	HarvWSNet	CEA/Leti research institute and University of Rennes, France	N/A	2013	N/A
3	UbiSec&Sens	Eurescom and NEC Europe Ltd., Germany	N/A	2009	http://www.ist-ubisecsens.org/
4	Emuli	Kent State University, USA	N/A	2007	N/A
5	MEADOWS	Hong Kong University of Science and Technology, China	N/A	2004	N/A
6	Freemote Emulator	University of Applied Science of Fribourg and University of Neuchâtel, Switzerland	9	20 October 2010	https://www.assembla.com/wiki/show/freemote
7	VMNet	Hong Kong University of Science and Technology, China	1.0.2	30 October 2005	http://www.cse.ust.hk/vmnet/
8	WSim	INRIA/Compsys and INRIA/ARES, France	N/A	4 January 2012	http://wsim.gforge.inria.fr/
9	EmStar	University of California, Los Angeles (UCLA), USA	2.5	October 2005	http://cens.ucla.edu/projects/2007/Systems/EmStar/
0	WiEmu	Arab Academy for Science and Technology, Egypt	N/A	18 Apr 2014	http://wiemu.sourceforge.net/apidocs/
1	WiSeREmulator	University of Houston, USA	N/A	2010	N/A
2	SUNSHINE	Virginia Polytechnic Institute and State University, USA	N/A	2011	http://rijndael.ece.vt.edu/sunshine/index.html
3	CORE	U.S. Naval Research Laboratory, USA	4.8	8 June 2015	http://www.nrl.navy.mil/itd/ncs/products/core

USN simulator and emulator features

This section outlines the features of the most prominent USN simulation and emulation environments and frameworks. In this context, 22 of the above 130 tools are selected to be compared in detail: those which have been supported by their developers, are popular, have published results, are usable, and have interesting characteristics and features. In this regard, Table 8 presents the features of the selected simulators and emulators, i.e., version type, license type, language/scripting used, open source capability, supported platform, and presence of on-line document or tutorial. Normally, the software is developed for academic, research, and commercial purposes. Academic and research versions of software are free of charge. For commercials, the licenses or affiliated packages are not provided for free for users (the underlined letter demonstrates the extensive use of the software in that version). There are two types of licenses: Gnu's Not UNIX General Public License (GNU GPL) which allows end users to use, study, share, and modify the software freely, and Berkeley Software Distribution (BSD) license, which imposes minimal restrictions on the redistribution of software. A variety of languages and scripts is employed by which the simulators/emulators are designed with, i.e., Java, C, C++, and NesC (Network embedded system C language), where each one has its own pros and cons. Open source software has the benefits of modifiable source code and free extension of the software. But, open source software may not be user-friendly and convenient with GUI. The platforms that simulators/emulators run on vary from Microsoft Windows to Linux and Mac OS. On-line documents, manuals and tutorials may help users to install, get started with, and overcome any difficulties while using the software.

Table 8. Specifications of USN simulators and emulators.

Simulator/Emulator	Version Type	License Type	Language/Scripting	Open Source	Platform	On-line Document /Tutorial
Prowler	r	Free	m-file	No	Linux, Windows, Mac OS	Yes
SENSE	r	Free (BSD)	C++, CompC++	Yes	Linux, Windows	Yes
Shawn	r	Free (BSD)	C++	Yes	Linux, Windows, Mac OS	Yes
JiST/SWANS	r	Free	Java, Jython	No	-	Yes
COOJA	r	Free (BSD)	Java	Yes	Linux	Yes
J-Sim	r	Free (BSD)	Java, Tcl	Yes	Linux, Windows, Mac OS, Solaris	Yes
Ptolemy II	r	Free	Java	Yes	Windows, Mac OS	Yes
SENS	r	Free (BSD)	C++, NesC	Yes	Linux	No
NS-2	arc	Free (Apache License v2, BSD,GNU GPL v2)	C++, OTcl	Yes	Linux, Solaris, SunOS, Windows, Mac OS	Yes
NS-3	ar	Free (GNU GPL v2)	C++, Python	Yes	Linux, Windows, Mac OS, Solaris	Yes

OMNeT++	arc	Free	C++, NED	Yes	Linux, Windows, Mac OS	Yes
Castalia	ar	Free (GNU GPL v2)	C++	Yes	Linux, Windows	Yes
OPNET	ac	Free (Academic) & Non-free (Commercial)	C, C++, Java	No	Linux, Windows	Yes
GloMoSim	r	Free	C, Parsec	Yes	Linux, Windows	No
QualNet	c	Non-free	C, C++	No	Linux, Windows, Mac OS	No
Worldsens	ar	Free	C	No	-	Yes
ShoX	r	Free (GNU GPL v2)	Java	No	Linux, Windows, Mac OS	No
Wireshark (Ethereal)	ar	Free (GNU GPL)	C, C++	Yes	Unix, Windows, Mac OS	Yes
TOSSIM	r	Free (BSD)	C++, NesC, Python	Yes	Linux, Windows	Yes
ATEMU	ar	Free (BSD)	C	Yes	Linux, Windows, Solaris	No
Avrora	r	Free (BSD)	Java	Yes	Linux, Windows, Mac OS	No
EmStar	r	Free	C	Yes	Linux, Windows	No

In addition, for each of the selected simulators and emulators, Table 9 shows whether they are general or specific purpose simulator/emulators, support GUI, are object-oriented or component-based, support the level of abstraction, including the energy consumption models, and if any derivative or extension is developed. Simulators and emulators are either general (adaptive) or specific (new) environments. Some of the simulators and emulators are GUI-driven where the user can drag and drop the network components, while others require command line and scripting. Object-oriented based simulators and emulators focus on the relationships between classes and facilitate implementation and extensibility, but they lack scalability. On the other hand, component-based simulators and emulators focus on interchangeable code modules that function independently; thus, they are more extensible and scalable, but may be more difficult to implement in a modularized way [39]. The simulation abstraction level is described in section 2.1.4. Alphabetical letters are used to briefly address each simulator and emulator abstraction category: generic network simulator, code level simulator, firmware level simulator, algorithm level simulator, packet level simulator, and instruction level simulator. Energy consumption models consider whether sensor nodes have batteries or produce energy for efficient system design.

Simulators and emulators derivatives or extensions are developed to improve the simulators/emulators performance and/or overcome any deficiency.

Table 9. Features of USN simulators and emulators.

Simulator/Emulator	General/Specific Purpose	UI	Object-oriented/Component-based	Simulation Level Abstraction	Energy Consumption Model	Derivative/Extension
Prowler	Specific	es	N/A	FP	No	JProwler
SENSE	General	o	Component-based	P	Yes	N/A
Shawn	Specific	o	N/A	C	Yes	N/A
JiST/SWANS	General	o	N/A	GFP	No	N/A
COOJA	General	o	N/A	GCF	Yes	SensEH
J-Sim	General	es	Component-based	P	Yes	G-JSim
Ptolemy II	General	o	N/A	F	Yes	Viptos, VisualSense
SENS	Specific	o	Component-based	GCP	Yes	N/A
NS-2	General	o	Object-oriented	GP	Yes	Mannasim, NRL Sensorsim, RTNS, SUNSET, Aqua-Sim, WOSS, TRAILS, PiccSIM
NS-3	General	o	Object-oriented	GP	Yes	Symphony
OMNeT++	General	es	Component-based	G	Yes	SENSIM, Castalia, MiXiM, NesCT, PAWiS
Castalia	General	es	N/A	GA	Yes	SolarCastalia

OPNET	General	es	Object-oriented	FP	Yes	-
GloMoSim	General	es	Object-oriented	GP	Yes	Aqua-GloMo, QualNet
QualNet	General	es	N/A	GAP	Yes	sQualNet, SenQ
Worldsens	Specific	es	N/A	P	Yes	WSim, WSNet
ShoX	Specific	es	Object-oriented	N/A	Yes	N/A
Wireshark (Ethereal)	General	es	N/A	N/A	No	N/A
TOSSIM	Specific	es	Component-based	CAI	Yes	JTOSSIM, mTOSSIM, PowerTOSSIM z, TOSSF, TYTHON, Mule
ATEMU	Specific	es	N/A	FI	No	N/A
Avrora	Specific	o	N/A	I	Yes	AvroraZ, AEON
EmStar	Specific	es	Component-based	GFI	Yes	N/A

Qualitative information facilitates the comparison of the simulator/emulator and the process of choosing an appropriate one among several tools. Such classification has more adherents among students since it enables them to compare and assess the tools relatively. In this context, Table 10 gives qualitative details of the selected USN simulation and emulation tools for academic, research, and commercial versions. The first criterion, visualization, is related to the environment where the user is manipulating his/her practices. Visualization ranges from poor to excellent qualities. Flexibility deals with how many frequent runs and configuration changes may be applied to a simulator/emulator. Scalability, as explained before, represents the number of nodes which a simulator/emulator may handle. Existing protocols in simulators/emulators databases are shown by protocol availability criterion. Radio signals propagate via an antenna from a sensor node. The strength of signals is directly related to the distance. Therefore, the formidable presence of such modules in USN simulators/emulators is an advantage. The degree of simplicity represents how quickly individuals get familiar with tools, while interactivity demonstrates how easily individuals interact and exploit tools. Such criteria are considerable for academic and educational purposes where increased simplicity and interactivity makes the tool desirable and pleasant [91].

Table 10. Qualitative specifications of selected USN simulators and emulators for academic, research, and commercial versions.

Simulator/Emulator	Visualization	Flexibility	Scalability	Protocol availability	Radio signal propagation model	Simplicity-Interaction (for academic version)
Prowler	Basic (r)	Medium (r)	Medium (r)	Small (r)	Basic (r)	N/A
SENSE	Average (r)	Medium (r)	Medium (r)	Medium (r)	Average (r)	N/A
Shawn	Average (r)	Medium (r)	High (r)	Small (r)	Average (r)	N/A
JiST/SWANS	Basic (r)	Medium (r)	Very high (r)	Medium (r)	Good (r)	N/A
COOJA	Good (r)	High (r)	Medium (r)	Small (r)	Good (r)	N/A
J-Sim	Average (r)	Medium (r)	Low (r)	Medium (r)	Average (r)	N/A
Ptolemy II	Good (r)	Medium (r)	Medium (r)	Small (r)	Good (r)	N/A
SENS	Average (r)	Medium (r)	Medium (r)	Small (r)	Good (r)	N/A
NS-2	Good (ar)	High (r)	Medium (rc)	Large (rc)	Good (rc)	Low-Medium
NS-3	Good (ar)	High (r)	High (r)	Medium (r)	Good (r)	Medium-Medium
OMNeT++	Excellent (ar)	High (r)	Medium (rc)	Medium (rc)	Good (rc)	Medium-Medium
Castalia	Excellent (ar)	High (r)	Medium (r)	Medium (r)	Good (r)	Medium-Medium
OPNET	Excellent (ar)	N/A	High (c)	Large (c)	Excellent (c)	High-High
GloMoSim	Good (r)	Medium (r)	High (r)	Medium (r)	Average (r)	N/A
QualNet	Good (r)	N/A	High (c)	Medium (c)	Good (c)	N/A
Wireshark (Ethereal)	Good (r)	Medium (r)	N/A	Large (r)	N/A	N/A
TOSSIM	Good (r)	Medium (r)	Medium (r)	Small (r)	Basic (r)	N/A

ATEMU	Average (ar)	Medium (r)	Low (r)	Small (r)	Basic (r)	Medium-High
Avrora	Poor (r)	High (r)	Medium (r)	Small (r)	Average (r)	N/A
EmStar	Good (r)	Medium (r)	Low (r)	Medium (r)	Average (r)	N/A

A simulator or emulator is designed or developed to fulfill the needs of a project or application. Therefore, we cannot generically and lucidly name a simulator/emulator as the perfect and flawless tool. What is desirable is to identify the positive and negative aspect of each and pick the one which best fits the application. In this regard, Table 11 summarizes several key features and the limitations corresponding to each simulation and emulation tool.

Table 11. Key features vs. limitations of USN simulators and emulators.

Simulator/Emulator	Key Features	Limitations
Prowler	-Rich library of radio modules and protocols -Extendable for general platforms -Easy prototyping of applications -Integration of different optimization algorithms -Provide GUI and good visualization capabilities -Good extensibility via plug-ins	-Provides only one TinyOS MAC protocol by default -No sensor node energy modeling -No 3D space simulations -No detailed antenna modeling in the package
SENSE	-Balanced between modeling methodology and simulation efficiency -Memory-efficient, extensible, scalable, and reusable -Supports parallel simulation -Offers different battery models -User-friendly and fast	-Not accurate evaluation of USN research -Lacks a comprehensive set of models, routing protocols and a wide variety of configuration templates for USN -Absence of GUI
Shawn	-Able to simulate large-scale USN -Able to select the application preferred behavior -Full access to the communication graph -Protocols can be modified -Easy to determine the effect of channel parameters	-Absence of GUI -MAC module is not extent -Lots of programming is required -Detailed simulations of issues like radio propagation properties or low-layer issues are not well considered -Simulation issues or lower layer issues are not considered -Limited to generate a postscript file

JiST/SWANS	-Supports parallel simulation -Enables the analysis of large-scale network behavior	-Lack of enough protocol models -Is a command-line-based simulator -Realization of specific application scenarios and the user interaction is difficult -Only focuses on static application scenarios -Absence of GUI
COOJA	-Considers both simulated hardware and software -Larger-scale behavior protocols and algorithms can be observed	-Not extremely efficient -Supports a limited number of simultaneous node types -Making extensive and time dependent simulations difficult -Lack of sensor network protocols and applications -Absence of GUI
J-Sim	-Simulate real-time processes -Simulate radio channels and power consumptions -The execution time is much longer -Provides support for energy modeling, with the exception of radio energy consumption -Support mobile wireless networks and sensor networks -Good reusability and interchangeability -Easy installation on different platforms -Specific packages with both battery and power model -Provides GUI -Lots of memory space	-Low efficiency of USN simulation -The only MAC protocol provided for wireless networks is 802.11. -Unnecessary run-time overhead in intercommunication model -Relatively complicated to use
Ptolemy II	-Provides Java packages that support different models of simulation paradigms -Models are constructed in an actor-oriented way, very similar to the component-based design	-Absence of GUI -Does not support protocols above wireless medium -Only support sound sensors

SENS	-Platform-independent -Users can assemble application-specific environments -Defines environment as a grid of interchangeable tiles	-Not accurately simulate a MAC protocol -Provides support for sensors, actuators, and physical phenomena only for sound -Does not support physical phenomena of sensors or environmental effects -less customizable -Absence of GUI
NS-2	-Easy to add new protocols -A large number of protocols available publicly -Extensible features -Object-oriented design allows creating and using of new protocol -Excellent extensibility	-Cannot simulate problems of the bandwidth or the power consumption in USN -Supports only two wireless MAC protocols, 802.11, and a single-hop TDMA protocol -Absence of GUI (employs visualization tool-NAM (Network Animator)) -Limited scalability (in memory usage and simulation run time) -Requires user scripting knowledge and experience
NS-3	-Supports simulation and emulation modes -Supports a real-time schedule -Ability to support multiple radio interfaces and multiple channels -Better scalability compared with NS-2 -A simulation script can be written as a C++ program, which is not possible in NS-2	-Some restrictions on the customization exist -Lack of an application model -Does not run real hardware code -Does not scale well for USN -Absence of GUI (employs a package known as PyViz, which is a python based real-time visualization package)
OMNeT++	-Provides a powerful GUI -Supports MAC protocols and some localized protocols -Simulate power consumptions and channel controls -Excellent extensibility -Fast processing time	-Lack of available protocols in its library -Most of the available models have been developed by independent research groups and do not share a common interface -Simple energy model
Castalia	-Physical process modeling, sensing device bias and noise, node clock drift, and several MAC and routing protocols implemented -Highly tunable MAC protocol and a flexible parametric physical process model	-Is not a sensor specific platform -Not useful if one would like to test code compiled for a specific sensor node platform

OPNET	-Free for academic use -Uses a hierarchical model to define each characteristic of the system -Capability of recording a large set of user defined results -Powerful GUI -Supports the use of modeling different sensor-specific hardware -Contains extensive libraries of accurate models -Easily extensible -Code is very well documented and ships with a large number of built-in protocols -Fast processing time	-Scalability problems -Very expensive license
GloMoSim	-Supports protocols designed purely for wireless networks -Built using a layered approach -Uses standard APIs between different simulation layers -Processing time is medium -Large scalability -Good mobility models specify for wireless simulation -Transport layer and IP address support -Parallel simulation capability -Supports ad-hoc networking protocols -GUI support	-Not scalable of simulating sensor networks accurately -Does not support phenomena occurring outside of the simulation environment -Only supports simulating IP networks -Unavailability of new protocols -No specific routing protocols for sensor network -No energy consumption models
QualNet	-A comprehensive set of advanced wireless modules is provided -User-friendly tool -Sophisticated animation capabilities -Support for multi-processor systems and distributed computing -Extensibility is good	-Annual license is expensive -Limited online resources and tutorials are available

Worldsens	-Supports large-scale USN simulation -Language dependent: runs native code generated for the target microcontroller -Enables accurate time control	-Node architecture is limited to systems based on MSP430 microcontroller from Texas Instruments and on RF transceivers from the same manufacturer -Co-simulation generates significant simulation time
ShoX	-GUI and visualization support -Architecture	-Simulator user guide and documentation is unavailable -Lack of models
Wireshark (Ethereal)	-Supports hundreds of protocols -Supports rich display filter capabilities -Packet sniffer – live capture and offline packet analysis	-Considerable protocol knowledge is need for deep analysis and inspection
TOSSIM	-Designed specifically for TinyOS applications to be run on MICA motes -Possible to build scalable and high fidelity simulations of full sensor network applications -Graphical User Support (Tiny ViZ) -Simple and powerful emulator for USN -Support thousands of Nodes -High degree of accuracy or running the application source code unchanged -Can emulate radio models and code executions -Good extensibility -Processing time is fast	-TOSSIM would be effective for simulating thread-based systems -The cost of the large number context switches (even if in user-land) would be prohibitive -Not good for low level protocols (MAC) -Does not simulate the physical phenomena that are sensed -Each node must run the exact same code -Makes several assumptions about the target hardware platform -Does not model energy consumption by itself (possible with add-on PowerTOSSIMz) -Assumes that each node in the network must run the exact same code, so making it less flexible -Unsuitable for heterogeneous environments
Avrora	-Instruction-level simulator -Provides fast speed and good scalability -Enables validation of time-dependent properties of large-scale networks -Supports energy consumption simulation -Can simulate different programming code projects	-Fails to model clock drift -50% slower than TOSSIM -Cannot model mobility -Absence of GUI -Cannot simulate network management algorithms

ATEMU	-One of the most accurate sensor simulators -Uses a cycle-by-cycle strategy to run application code -Can simulate multiple sensor nodes at the same time -Has a large library of a wide range of hard devices -Can provide a very high level of detail emulation in USN -GUI can help users debug programs	-Scalability problems -Long simulation time -Has fewer functions to simulate routing and clustering problems
EmStar	-Support modular programming model -Can mitigate faults among sensors -Evaluation of bugs is much easy -GUI support available -Fast processing time -User friendly -Supports hybrid mode -Provides an option to interface with actual hardware while running a simulation -Compatible with two different types of node hardware	-Limited scalability -Only run in a real time simulation -Supports only the code for the types of nodes that it is designed to work with

Performance assessment of simulators and emulators

Selecting the most appropriate simulator or emulator for USN purpose among the numerous versions available remains a controversial task. Several studies have evaluated the performance of USN simulators and emulators and analyzed and compared the results with each other in terms of popularity, architecture, OS, credibility, accuracy, scalability, execution speed and time, CPU usage, visualization and GUI, debugging, and even learning difficulty criteria. Each study has defined a scenario comprised of several parameters. In this context, Table 12 addresses these efforts in brief.

Table 12. Comparative studies of the performance of USN simulators and emulators.

Reference	Compared Simulators/Emulators	Scenario Parameters	Performance assessment
[304]	NS-2, NS-3, GloMoSim, and OMNet++	Simulation Time: 500 s X, Y Dimensions: 1000 x 1000 Mobility Model: None Packet size: 512 kb	Number of nodes vs. Memory usage Number of nodes vs. CPU utilization Number of nodes vs. Computational time

		Number of nodes: 400-2000 Routing protocol: AODV	
[75]	NS-2, TOSSIM, and Shawn	Simulation Time: 60 s Rate of sending packet: 250 ms X, Y Dimensions: 500 x 500 Number of nodes: 10000	Number of nodes vs. Abstraction level Number of nodes vs. CPU time Number of nodes vs. Memory usage
[74]	Castalia, MiXiM, and TOSSIM	Number of nodes: 15	Packet Reception Rate (PRR) for different log normal shadowing settings in Castalia PRR for different noise floor in Castalia PRR for different modulation techniques in Castalia PRR for different deciders in MiXiM PRR for different noise floor in TOSSIM PRR for different noise models in TOSSIM
[305]	NS-2, OMNeT++, and OPNET	Application: Fire fighter Simulation Time: 500 s Rate of sending packet: 0.2 s Node speed: 0.5 km/h Number of nodes: 25 Packet size: 32 bytes Routing protocol: AODV	Throughput and delay at firefighter Throughput and delay at command post node Received throughput at fire fighter node Received throughput at incident commander node Firefighter to incident commander packet delay Firefighter to incident commander packet delay frequency distribution
[306]	OPNET, GloMoSim, and NS-2	Terrain size: 1km x 1km Number of nodes: 50 Node placement: uniform No. of broadcasting nodes: 10 No. of broadcasts per node: 100 MAC protocol: 802.11 without RTS/CTS Bit rate: 2 Mbps Wireless propagation model: FreeSpace Antenna Type: Omni directional	Success rate vs. Power range Success rate vs. Mobility Overhead vs. Mobility Time delay vs. Mobility

		Mobility model: Random waypoint Minimum node speed: 0 m/s	
[240]	IDEA1 and NS-2	Number of nodes: 31 Simulation Time: >1000 s	Accuracy evaluation, simulation time, and power dissipation analysis
[184]	SXCS and OMNet++	Number of nodes: 10-1000	Agents processing time vs. Number of nodes Remaining energy profiling Memory usage vs. Number of nodes Packet loss vs. Number of nodes
[307]	NS-2, NS-3, OMNet++/Castalia, TOSSIM, and J-Sim	Simulation Time: 500 s X, Y Dimensions: 1000 x 1000 Mobility Model: None Packet size: 512 kb Number of nodes: 400-2000 Routing protocol: LEACH	Number of nodes vs. Memory usage Number of nodes vs. CPU utilization Number of nodes vs. Computational time
[308]	NS-2 and JiST/SWANS	Number of nodes: 200-1000 Transmission range: 250 m Field: 2368-5296 m Mobility: Random waypoint Max speed: 20 m/s Min speed: 1 m/s Pause: 0s Duration: 120 s Warm up: 20 s Cool down: 10 s Noise: Independent Path loss: Tworay Fading: None Packet loss: None Traffic: 1 packet/min	Number of nodes vs. Delivery success ratio Number of nodes vs. Hopcount of message transfer Number of nodes vs. Average message delay Number of nodes vs. Processing time of CGGC routing protocol Number of nodes vs. Processing time of AODV routing protocol Number of nodes vs. Memory usage

		Routing protocol: CGGC, AODV Beaconing: 1 Hz Packet caches: Unlimited Destination radius: 100-300 m	
9]	[30	JiST/SWANS and OMNet++/INETMANET	VANET scalability: Circular road and rectangular road Number of Vehicles: >5000 Execution times: 3-10 Routing protocol: AODV	Number of vehicles vs. Overall time for simulations Number of vehicles vs. Memory consumption
0]	[31	OMNeT++, NS-2, and OPNET	Number of nodes: 500-2000 X, Y Dimensions: 500 x 500 m Simulation time: 300 s Query generating nodes: 10 and 100	Comparison of delivery ratio Execution time of 10 queries generate nodes (SimpleMAC) Execution time of 100 queries generate nodes (SimpleMAC) Memory usage of 10 queries generate nodes (SimpleMAC) Memory usage of 100 queries generate (SimpleMAC) Execution time of 10 queries generate nodes (IEEE 802.11MAC) Execution time of 100 queries generate nodes (IEEE 802.11 MAC) Memory usage of 10 queries generate nodes (IEEE 802.11 MAC) Memory usage of 100 queries generate nodes (IEEE 802.11 MAC)
0]	[15	NS-2 and J-Sim	Number of sink nodes: 1 Number of target nodes: 2 Number of sensor nodes: $n^2 - 1$ X, Y Dimensions: 1500 × 1500 m Target nodes speed: 10 m/s Nodes' sensing radius: 200 m Simulation time: 1000 s Routing protocol: AODV (Scenario A) Routing protocol: GPSR (Scenario B)	Network size $n^2 + 2$ vs. Execution time Network size $n^2 + 2$ vs. Number of events Memory usage before simulation start vs. Network size $n^2 + 2$ Memory usage before simulation ending vs. Network size $n^2 + 2$ Execution time vs. Network size $n^2 + 2$ (GPSR routing protocol) Number of events vs. Network size $n^2 + 2$ (GPSR routing protocol) Memory usage before simulation start vs. Network size $n^2 + 2$ (GPSR routing protocol) Memory usage before simulation ending vs. Network size $n^2 + 2$ (GPSR routing protocol)
1]	[31	TOSSIM, TimeTOSSIM, and Avrora	Number of nodes: 650 Simulation time: <1000 s	Scalability comparison: Number of nodes vs. Time Accuracy, speed, and energy consumption

[118]	NS-2 and LSU SensorSimulator	Number of nodes: 5-200 MAC layer: 802-11 MAC	Number of nodes vs. Delivery ratio Execution time for 10 queries for 150 simulation seconds Execution time for 100 queries for 150 simulation seconds Memory utilized to setup the network for 10 queries Memory utilization during simulation for 10 queries Memory utilized to setup network for 100 queries Memory utilization comparison during simulation for 100 queries Directed Diffusion-GEAR-MAC802.11 execution time for 10 queries simulated for 300 simulation seconds Directed Diffusion-GEAR-MAC802.11 memory usage for 10 queries simulated for 300 simulation seconds

One approach for conducting a relative comparison of USN simulation/emulation performance is to define scenario(s). Each scenario is comprised of a set of parameters to run. These parameters may vary with respect to simulators/emulators throughput and defined application; however, such tool assessment has some general parameters in common. We categorize these parameters into four groups of node, execution, terrain, and other, as presented in Figure 7. The node category comprises several qualitative and quantitative features, which are attributed to the sensor nodes, ranging from sensor numbers to sensor functionalities. The execution category is dedicated to all the settings for carrying out a simulation/emulation, such as simulation time, packet characteristic, protocol type, etc. The terrain (field) category is one of the most important components in simulation/emulation. In real-world applications the sensed data from the sensors need to be routed to the targeted sensors/services. Any physical intruder (e.g., wall, topography) that cause problem in proper transmission of these data needs to be simulated in advance. Finally, there are several general and specific components that do not fall into the earlier categories, which are placed in other category.

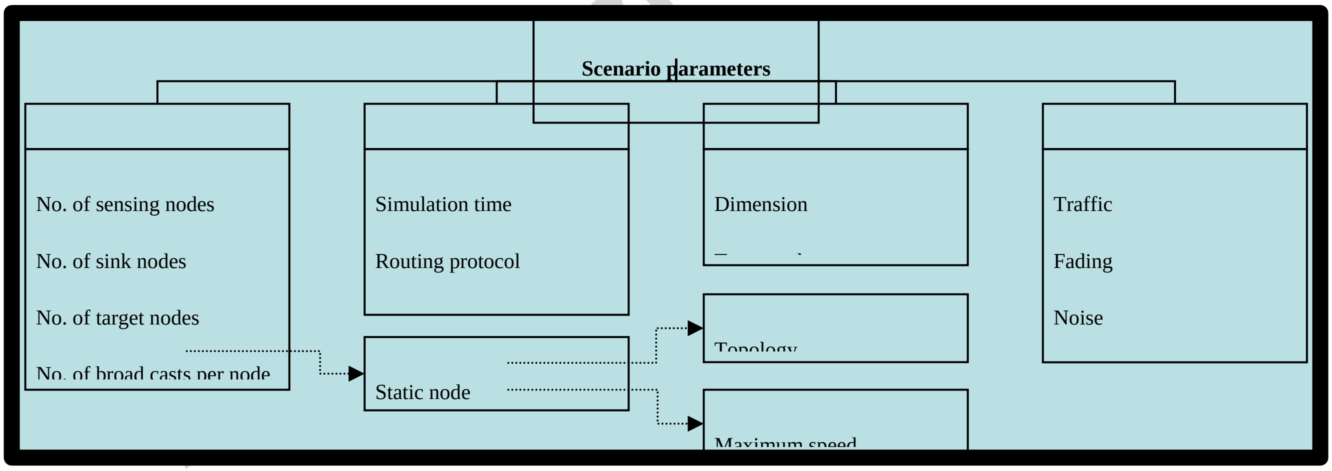

Figure 7. Scenario parameters for USN simulation and emulation.

Although Table 3 has proposed several features for simulation assessment, the most critical one, i.e., scalability (or number of nodes), needs to be assessed specifically for USN simulators and emulators. This feature demonstrates the ultimate throughput of a simulator/emulator in handling a number of sensor nodes effectively. Scalability is so imperative that the majority of comparative studies in Table 12 have compared it against other performance assessment features (see Figure 8). A review of the literature acknowledges that the number of sensor nodes directly affects other features, as well as the final performance of the simulator/emulator.

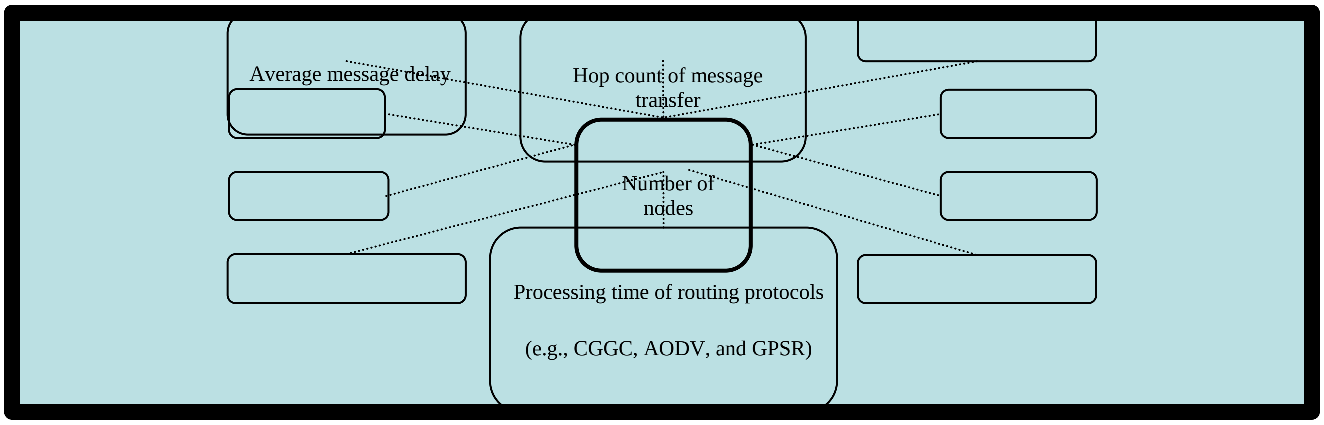
Average message delay
Hop count of message transfer
Number of nodes
Processing time of routing protocols
(e.g., CGGC, AODV, and GPSR)

Figure 8. Pairwise performance assessment: number of nodes feature versus other simulator's/emulator's features.

Scalability and the potential number of sensor nodes deployed in simulation and emulation environments have been challenging tasks. Given the widespread and pervasive projects of USN and emerging technologies in USN, simulation and emulation by more sensor nodes can be an advantage. However, a review of the literature ascertains that higher scalability and more sensor nodes heavily increase the execution time, CPU utilization, and memory usage, which affect the delivery success ratio of messages and in some cases delay the message. Figure 9 demonstrates the approximate number of sensor nodes that a simulator and emulator can handle effectively.

Simulator & Emulator

	WISE NSE
	OPN ET
	SENS
	Avror a
	OMN eT++
NS-2	sQual Net
Visual Sense	SWA NS
Ptolem y II	TOSS F

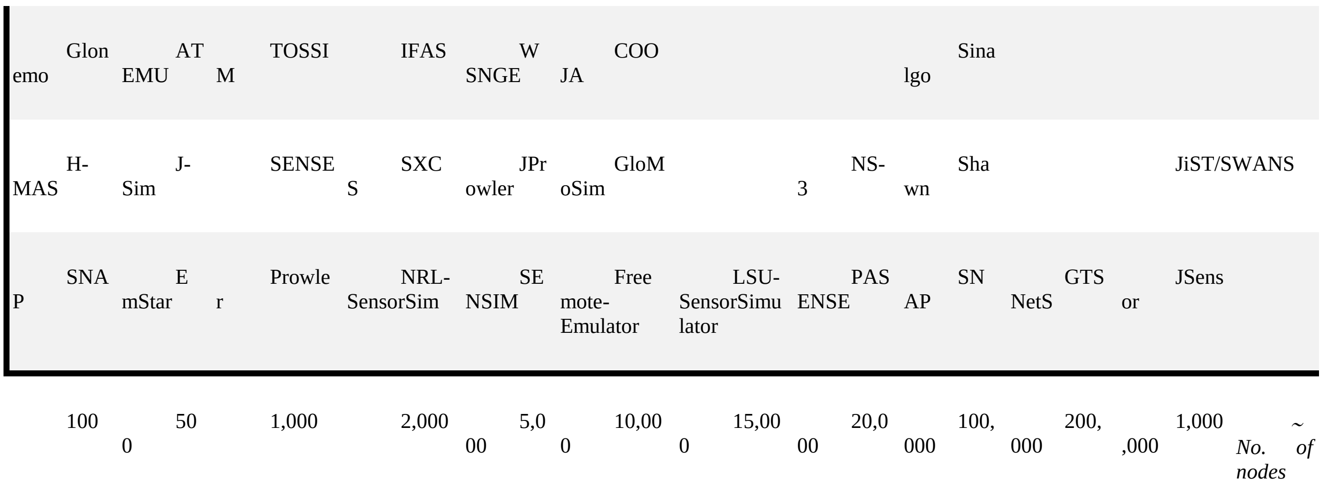

Figure 9. Simulators’ and emulators’ scalabilities: approximate number of supported sensor nodes.

Simulators’ and emulators’ designers and developers have employed a variety of programming languages for generating such tools. In this research, the programming languages of 102 out of 130 simulators and emulators were found in their corresponding websites and tutorials as well as the articles that firstly introduced the tools. A review of literature demonstrates that almost half of the simulators and emulators are scripted by C/C++ and their derivatives (e.g., C++ Builder, CompC++, NesC, PARSEC), while 39 percent have made use of Java programming language. In this regard, C/C++ engines are expected to have better functionality and productivity than their Java counterparts. Python is used for developing 6 percent of simulators and emulators. A small proportion of simulators or emulators have utilized other programming languages (e.g., C# and m-file). This information is illustrated as a chart in Figure 10.

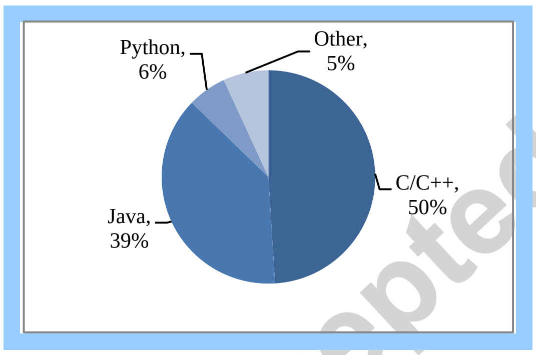

Figure 10. Percentage of simulator/emulator programming languages.

Lesson Learned, Open Issues, and Future Directions

In this section, we first summarize the lesson learned from literature, and then point out some general and specific research directions for USN simulators and emulators.

Lesson Learned

This survey has presented several lessons. We have highlighted some tips for students and researchers, for whom this paper has been targeted at. The study should clarify the choice of a simulator or an emulator. The goal of design and development of a simulator/emulator varies among the available versions. Each tool has its own advantages and disadvantages. So, choose the one that best fits your application, effort, time, and budget. Most of the presented USN simulators and emulators fall into research and commercial domains. Among the simulators, NS-2, OPNET, and QualNet, and for the emulators, TOSSIM, have gained more popularity. From an educational perspective, limited tools are available that have an academic version released. Among them, OPNET IT Guru and OMNet++ have gained more attention, especially due to their free license, rich tutorial, and excellent GUI. A good GUI facilitates interaction between users and software by dragging and dropping the simulation elements and presenting the outputs graphically. However, if a student neglects the GUI and can get familiar with scripting, NS-2 and NS-3 are suitable educational tools. Last but not least, an open source facility enables modification of current programs and free extension of the software. However, they may not be user-friendly and convenient with GUI. NS-2 and OMNeT++ are two frequently used open source software.

Choosing an authoritative tool that provides flexible modeling and validation can drastically improve the results. Also, the tool should enable statistical analysis of output data. Users/developers should ensure the validity of the model inputs and outputs. Researchers and tool developers should consider the positive and negative aspects of simulators/emulators, appropriate programming language, architecture (i.e., component-based or object-oriented architecture), degree of simulator/emulator complexity, presence and shortage of features, parallel execution, real deployment of sensor nodes, and several other factors.

Researchers normally execute simulation/emulation repeatedly by using one simulator/emulator. It should be noted that execution of several runs does not necessarily results in better results. The simulation/emulation outputs are directly related to the mathematical models developed in that specific tool. The variation in the built models leads to discrepancies among the simulation/emulation outputs. Although simulation/emulation models should be built credibly, more complex models require more computational time and resource. A good USN simulator/emulator, however, offers a balance between several criteria, such as accuracy, scalability, feature, extendibility, scripting language, GUI support, and ease of use.

Future Work

Many types of research have been conducted addressing simulation and emulation issues in USN. However, there are still a lot of potential future studies which either remain unsettled or unexplored comprehensively. We classify them into general and specific open issues. From the general perspective we have identified the following trends.

A promising future work relates to deeply reviewing other performance evaluation techniques (i.e., analytical modeling, physical testbed, and real world experimentation) suitable for USNs along with addressing a standardized criteria list for assessing the performance of such techniques.

Interoperability of USN simulators/emulators is a topic that has not been deeply explored so far; that is, developing integrated simulators/emulators to support a wide range of features. In this context, one tool can be complemented by other tools for their distinctive features such that the output of one can be imported as an input to the other one. To this end, a model can be analyzed through different simulation tools synchronously/asynchronously. This enables the strengths and weaknesses of the model to be revealed and makes the model possible to be improved in the design process. Experimenting with multiple simulators/emulators is a non-trivial challenge and should be supported with a common API for all participants.

USN has gained attention in different indoor and outdoor applications, such as health, transportation, agriculture, and military, to name a few [312-316]. These applications have specific characteristics that are coupled with technology. Therefore, there is an enormous potential to run and test application-specific scenarios through USN simulators and emulators. Since simulations and emulations can reveal design flaws to a large extent, the scenarios need to compare different parameters to increase the knowledge about impacting factors.

Given the ability of simulators/emulators in modeling sensor networks in diverse scales, exploring their capabilities in terms of terrestrial, underground, underwater, multi-media, and mobile USNs is an interesting area of research that can be carried out in future.

Besides the aforementioned general directions, we also highlight the following specific issues, which are not fully addressed or remain unaddressed by the current USN simulators and emulators so far. A promising future direction can be toward extending the functionality of simulators and emulators, especially the open-source ones, which suffer from the appropriate extension/feature of the USN emerging technologies (e.g., cognitive radio sensor networks, the Internet of Things (IoT), cloud computing, etc. [317]).

Cognitive radio sensor networks: In applications that require a large number of sensor nodes, the available bandwidth may not suffice to support all the transmissions, which can result in loss of useful data. In order to minimize data loss, an emerging trend in USNs is to equip the wireless sensor nodes with cognitive radio (CR) technology. CR is an adaptive, intelligent radio and network technology, capable of automatically detecting vacant channels (termed spectrum sensing) in a wireless spectrum, change their transmission parameters accordingly (termed spectrum decision), and making use of these available channels in an opportunistic manner, improving the overall spectrum utilization [318]. Incorporating cognitive radio capability in sensor networks yields a new sensor networking paradigm, termed cognitive radio sensor networks (CRSNs). Depending on the application, a USN composed of sensor nodes equipped with cognitive radio may benefit from its potential advantages. Several researches have investigated the theories behind this technology, such as radio resource allocation in CRSNs [319], channel bonding in CRSNs [320], and adaptive medium access control in CRSNs [321], to name a few. In spite of these considerations, the majority of researchers are using analytical methods to understand the behavior of CRSNs and a very few simulation models at present are providing support for combined features of USNs and CRNs. Accordingly, CRSNs has not been a main research focus in USN simulators and emulators. As an exceptional and pioneering study, a NS-2 based CRSN simulator model was proposed by [322]. This model supports the fundamental requirements of a CR-based wireless sensor network. As the research trend is shifting towards CRSNs technology, there is a possibility of examining different aspects of such technology and practically identifying the challenges in multiple applications (e.g., indoor sensing, multimedia, multiclass heterogonous sensing, real-time surveillance, etc.) by USN simulators and emulators for future research.

Energy management and harvesting: In USN applications static and dynamic sensor nodes are normally dispersed over a large area while they are prone to failure due to energy depletion. Exploiting power suppliers from fixed utilities may not be technically or economically possible in all practices. Therefore, energy management, harvesting, and replenishment become crucial to maximizing sensor networks' lifetimes and throughputs [323, 324]. Despite the plethora of theories relevant to these issues, very few researches have pointed out the difficulties in energy efficient protocol design using simulators/emulators; for example, [325] analyzed and compared a limited number of routing protocols for energy harvesting USNs in different scenarios by the Castalia simulator. Therefore, measuring the energy consumption of the (mobile) sensor nodes and experiencing energy efficient protocols are topics that have not been deeply explored in simulators/emulators thus far. More specifically, energy saving mechanisms (e.g., energy-efficient routing protocol, power-conserving MAC protocol, battery management, energy-efficient packet scheduling, etc. [326]) besides renewable energy resources (e.g., light, vibration, heat, radio frequency, wind, etc. [324]) are the issues that can be individually or collaboratively investigated by different applications in USN simulators and emulators.

The Internet of Things: During recent years, a lot of attention has been towards establishing infrastructures for smart and context-aware environments [327]. In this respect, in the Internet of Things (IoT) paradigm, concurrent collection of data from numerous devices and communications between them has been a challenging task for network technologies. To overcome this challenge, sensor networks try to complement the sensing and communication infrastructures. The integration of sensor networks and IoT is theoretically discussed in [328]. Also, the review by [15] demonstrates the utility of IoT and USN integration in real-world applications. Therefore, in the light of USN simulators and emulators, exploring the scalability and topology issues along with communication protocols, targeting at IoT applications, can be an interesting topic for prospective research.

Cloud computing: For ubiquitous applications, the collected data by sensor nodes need to be available at any time, at any place. However, due to the limitations of sensors in storage, bandwidth, battery power, processing, security, etc., one drastic solution for communication among sensor nodes is using cloud infrastructure. High-performance computing, less maintenance, seamless availability, and scalability are only a few features of cloud computing. Therefore, combination of USNs with clouds enables sharing and analyzing real time sensor data pervasively on-the-fly. A USN-Cloud platform normally comprises of USN, cloud infrastructure, and client(s). USN consists of physical wireless sensor nodes to sense the environment and route the data to the cloud. The cloud provides the client(s) on-demand data/service over the Internet. Despite the effectiveness of USN-Cloud computation, very few works have undertaken specific features of it. [329] presented a model by combining the concept of USNs with the cloud computing paradigm, and demonstrated that how both can benefit from this combination. [330] proposed a novel framework to integrate the cloud computing paradigm with USN, aiming to facilitate the shift of data from USN to the cloud environment. In this perspective, more general models must be further developed and be evaluated in different scenarios to measure the advantages and shortcomings of USN-Cloud combination. A large part of the work can be handled in USN simulators and emulators through developing patches/extensions. Then, the output of simulators/emulators can be imported as input for cloud computing.

Conclusion

The purpose of this study was to expand the understanding of researchers and tool developers about the available simulation and emulation tools for USN. We believe this study will aid them in choosing an appropriate simulator and/or emulator for sensor network performance testing based on their requirements and constraints. In this context, this paper overviewed 130 simulator and emulator environments and frameworks that were originally designed or adapted for USN. The brief explanation provided for each tool accompanied by corresponding designer/developer, the latest version, and the software link. A number of prominent USN simulators and emulators were qualitatively and quantitatively compared based on multifarious criteria: those that are available for the community and supported by their developers, are popular, have published results, and have interesting characteristics and features. The strengths and weaknesses of each were comparatively addressed as well. Several studies that cited relative performance analysis of simulators and emulators were introduced. Finally, we summarized with some recommendations for potential future works. We conclude from the observations that choosing an appropriate simulator/emulator and building a correct simulation model are two important aspects for USN. However, there is no standard simulator or emulator for all USN applications; its choice depends on the operational environment.

Acknowledgments

This work was supported by an Inha University research grant.

References

1. Weiser, M., *Hot topics-ubiquitous computing.* Computer, 1993. 26(10): p. 71-72.

2. Agrawal, R., et al. *Fast similarity search in the presence of noise, scaling, and translation in time-series databases*. in *VLDB '95: Proceeding of the 21th International Conference on Very Large Data Bases*. 1995. Morgan Kaufmann Publishers Inc.

3. Weiser, M., *Some computer science issues in ubiquitous computing.* Communications of the ACM, 1993. 36(7): p. 75-84.

4. Weiser, M., *Ubiquitous computing.* Computer, 1993(10): p. 71-72.

5. Keefe, D. and A. Zucker, *Ubiquitous computing projects: A Brief history*. Ubiquitous Computing Evaluation Consortium, SRI, 2003: p. 1-19.

6. Rahman, M.A., A. Pakštas, and F.Z. Wang, *Network modelling and simulation tools.* Simulation Modelling Practice and Theory, 2009. 17(6): p. 1011-1031.

7. Patil, A. and P.M. Hadalgi, *Evaluation of Discrete Event Wireless Sensor Network Simulators.* International Journal of Computer Science and Network, 2012. 1(5).

8. Egea-Lopez, E., et al. *Simulation tools for wireless sensor networks*. in *Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'05)*. 2005.

9. Ge, M., et al., *Survey on key revocation mechanisms in wireless sensor networks.* Journal of Network and Computer Applications, 2016. 63: p. 24-38.

10. Kiess, W. and M. Mauve, *A survey on real-world implementations of mobile ad-hoc networks.* Ad Hoc Networks, 2007. 5(3): p. 324-339.

11. Dwivedi, A., V. Patle, and O. Vyas, *Investigation on Effectiveness of Simulation Results for Wireless Sensor Networks.* Information Processing and Management, 2010: p. 202-208.

12. Shen, H. and G. Bai, *Routing in wireless multimedia sensor networks: A survey and challenges ahead.* Journal of Network and Computer Applications, 2016. 71: p. 30-49.

13. Jevtić, M., N. Zogović, and G. Dimić. *Evaluation of wireless sensor network simulators*. in *Proceedings of the 17th Telecommunications Forum (TELFOR 2009)*. 2009. Citeseer.

14. Curiac, D.-I., *Towards wireless sensor, actuator and robot networks: Conceptual framework, challenges and perspectives.* Journal of Network and Computer Applications, 2016. 63: p. 14-23.

15. Rashid, B. and M.H. Rehmani, *Applications of wireless sensor networks for urban areas: A survey.* Journal of Network and Computer Applications, 2016. 60: p. 192-219.

16. Kim, B. and J.-H. Kim, *PASENS: Parallel Sensor Network Simulator*, in *Advanced Methods, Techniques, and Applications in Modeling and Simulation*. 2012, Springer. p. 15-24.

17. Akyildiz, I.F., et al., *Wireless sensor networks: a survey.* Computer networks, 2002. 38(4): p. 393-422.

18. Yick, J., B. Mukherjee, and D. Ghosal, *Wireless sensor network survey.* Computer networks, 2008. 52(12): p. 2292-2330.

19. Leelavathi, G., et al., *Design Issues on Software Aspects and Simulation Tools for Wireless Sensor Networks.* International Journal of Network Security & Its Applications (IJNSA), 2013. 5(2): p. 47-64.

20. Imran, M., A.M. Said, and H. Hasbullah. *A survey of simulators, emulators and testbeds for wireless sensor networks*. in *International Symposium in Information Technology (ITSim)*. 2010. IEEE.

21. Krop, T., et al. *JiST/MobNet: combined simulation, emulation, and real-world testbed for ad hoc networks*. in *Proceedings of the second ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*. 2007. ACM.

22. Farooq, M.O. and T. Kunz, *Wireless sensor networks testbeds and state-of-the-art multimedia sensor nodes.* Applied Mathematics & Information Sciences, 2014. 8: p. 935-940.

23. Chawda, R.K. and A. Dwivedi, *Quintessence of Existing Testbeds for Wireless Sensor Networks.* International Journal of Engineering Trends and Technology, 2014. 12(2): p. 186-192.

24. Steyn, L.P. and G.P. Hancke, *A survey of wireless sensor network testbeds*, in *IEEE Africon'11-The Falls Resort and Conference Centre*. 2011, IEEE. p. 1-6.

25. Kim, H., et al., *Experimental Research Testbeds for Large-Scale WSNs: A Survey from the Architectural Perspective.* International Journal of Distributed Sensor Networks, 2015. 2015: p. 18.

26. El-Darymli, K. and M.H. Ahmed, *Wireless Sensor Network Testbeds: A Survey*, in *Wireless Sensor Networks and Energy Efficiency: Protocols, Routing and Management: Protocols, Routing*

and Management, N. Zaman, K. Ragab, and A. Bin Abdullah, Editors. 2012, Information Science Reference. p. 148-205.

27. Horneber, J. and A. Hergenroder, *A Survey on Testbeds and Experimentation Environments for Wireless Sensor Networks.* Communications Surveys & Tutorials, IEEE, 2014. 16(4): p. 1820-1838.

28. Halder, S. and A. Ghosal, *A survey on mobility-assisted localization techniques in wireless sensor networks.* Journal of Network and Computer Applications, 2016. 60: p. 82-94.

29. Chen, C.-C., S.B. Nagl, and C.D. Clack, *A method for validating and discovering associations between multi-level emergent behaviours in agent-based simulations*, in *Agent and Multi-Agent Systems: Technologies and Applications*. 2008, Springer. p. 1-10.

30. Wooldridge, M., *An introduction to multiagent systems*. 2009: John Wiley & Sons.

31. Banks, J., *Handbook of simulation*. 1998: Wiley Online Library.

32. Mishra, S., et al., *Simulation in Wireless Sensor Networks.* International Journal of Electronics Communication and Computer Technology, 2012. 2(4): p. 176-182.

33. Madani, S.A., J. Kazmi, and S. Mahlknecht, *Wireless sensor networks: modeling and simulation*. 2010, Vienna University of Technology, Vienna, Austria.

34. Jain, R., *The art of computer systems performance analysis*. 2008: John Wiley & Sons.

35. Peacock, J.K., J.W. Wong, and E.G. Manning, *Distributed simulation using a network of processors.* Computer Networks, 1979. 3(1): p. 44-56.

36. ETH, D.C.G., *Sinalgo-simulator for network algorithms*. 2008.

37. Xu, J. and M.J. Chung, *Predicting the performance of synchronous discrete event simulation.* IEEE Transactions on Parallel and Distributed Systems, 2004. 15(12): p. 1130-1137.

38. Shu, L., et al., *NetTopo: A framework of simulation and visualization for wireless sensor networks.* Ad Hoc Networks, 2011. 9(5): p. 799-820.

39. Singh, C.P., O. Vyas, and M.K. Tiwari. *A survey of simulation in sensor networks*. in *International Conference on Computational Intelligence for Modelling Control & Automation*. 2008. IEEE.

40. Diallo, O., et al., *Simulation framework for real-time database on WSNs.* Journal of Network and Computer Applications, 2014. 39: p. 191-201.

41. Carley, T.W., *Sidh: A wireless sensor network simulator*. 2005.

42. Eriksson, J., *Detailed simulation of heterogeneous wireless sensor networks*, in *Department of Information Technology*. 2009, Uppsala University: SWEDEN. p. 100.

43. Du, W., et al. *Towards a taxonomy of simulation tools for wireless sensor networks*. in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*. 2010. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

44. Banks, J. *Selecting simulation software*. in *Proceedings of the 23rd Conference on Winter Simulation*. 1991. IEEE Computer Society.

45. Christhu raj, M.R., et al., *A Comprehensive Overview on Different Network Simulators.* International Journal of Engineering and Technology, 2013. 5(1): p. 325-332.

46. Sohraby, K., D. Minoli, and T. Znati, *Operating Systems for Wireless Sensor Networks*, in *Wireless Sensor Networks: Technology, Protocols, and Applications*. 2007, John Wiley & Sons. p. 273-282.

47. Farooq, M.O. and T. Kunz, *Operating systems for wireless sensor networks: A survey.* Sensors, 2011. 11(6): p. 5900-5930.

48. Fröhlich, A.A. and L.F. Wanner, *Operating system support for wireless sensor networks.* Journal of Computer Science, 2008. 4(4): p. 272-281.

49. Phani, A.M.R.V.A., D.J. Kumar, and G.A. Kumar, *Operating systems for wireless sensor networks: A survey technical report*, in *Department of Computer Science and Engineering*. 2007, Indian Institute of Technology Madras: Chennai, India.

50. Dwivedi, A., M. Tiwari, and O. Vyas, *Operating Systems for Tiny Networked Sensors: A Survey.* International Journal of Recent Trends in Engineering, 2009. 1(2): p. 152-157.

51. Dong, W., et al., *Providing OS support for wireless sensor networks: challenges and approaches.* Communications Surveys & Tutorials, IEEE, 2010. 12(4): p. 519-530.

52. Dong, W., et al., *Senspire OS: A predictable, flexible, and efficient operating system for wireless sensor networks.* Transactions on Computers, IEEE 2011. 60(12): p. 1788-1801.

53. Merrett, G.V., et al. *Energy-Aware Simulation for Wireless Sensor Networks*. in *6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*. 2009.

54. Zhu, C., et al., *A survey on coverage and connectivity issues in wireless sensor networks.* Journal of Network and Computer Applications, 2012. 35(2): p. 619-632.

55. Mekni, M. and B. Moulin. *A survey on sensor webs simulation tools*. in *Second International Conference on Sensor Technologies and Applications, SENSORCOMM'08*. 2008. IEEE.

56. Dwivedi, A. and O. Vyas, *An exploratory study of experimental tools for wireless sensor networks.* Wireless Sensor Network, 2011. 3(07): p. 215-240.

57. Musznicki, B. and P. Zwierzykowski, *Survey of simulators for wireless sensor networks.* International Journal of Grid and Distributed Computing, 2012. 5(3): p. 23-50.

58. Dwivedi, A. and O. Vyas, *Recent Developments in Simulation Tools for WSNs: An Analytical Study*, in *Simulation Technologies in Networking and Communications*, A.-S.K. Pathan, M.M. Monowar, and S. Khan, Editors. 2014, CRC Press: Boca Raton, FL. p. 495-518.

59. Karl, M. *A comparison of the architecture of network simulators ns-2 and tossim*. in *Proceedings of Performance Simulation of Algorithms and Protocols Seminar*. 2005. University of Stuttgart.

60. Khan, M.Z., et al. *Limitations of simulation tools for large-scale wireless sensor networks*. in *IEEE Workshops of International Conference on Advanced Information Networking and Applications (WAINA)*. 2011. IEEE.

61. Stetsko, A., M. Stehlik, and V. Matyas. *Calibrating and comparing simulators for wireless sensor networks*. in *8th International Conference on Mobile Adhoc and Sensor Systems (MASS)*. 2011. IEEE.

62. Nayyar, A. and R. Singh, *A Comprehensive Review of Simulation Tools for Wireless Sensor Networks (WSNs).* Journal of Wireless Networking and Communications, 2015. 5(1): p. 19-47.

63. Khemapech, I., A. Miller, and I. Duncan, *Simulating wireless sensor networks*, in *School of Computer Science*. 2005, University of St Andrews: Scotland, UK. p. 10.

64. Egea-Lopez, E., et al., *Simulation scalability issues in wireless sensor networks.* Communications Magazine, IEEE, 2006. 44(7): p. 64-73.

65. Neves, P., J. Fonsec, and J. Rodrigue. *Simulation tools for wireless sensor networks in medicine: a comparative Study*. in *International Joint Conference on Biomedical Engineering Systems and Technologies, Funchal*. 2007.

66. Köksal, M.M., *A survey of network simulators supporting wireless networks*, in *Department of Computer Science*. 2008, Middle East Technical University: Ankara, Turkey. p. 1-11.

67. Lessmann, J., et al. *Comparative study of wireless network simulators*. in *Seventh International Conference on Networking (ICN 2008)*. 2008. IEEE.

68. Kuorilehto, M., M. Hännikäinen, and T.D. Hämäläinen, *Rapid design and evaluation framework for wireless sensor networks.* Ad Hoc Networks, 2008. 6(6): p. 909-935.

69. Wei, D., *On the Simulation of Networked Sensor Systems.* p. 1-8.

70. Mehta, S., et al. *A case study of networks simulation tools for wireless networks*. in *Third Asia International Conference on Modelling & Simulation (AMS'09)* 2009. IEEE.

71. Korkalainen, M., et al. *Survey of wireless sensor networks simulation tools for demanding applications*. in *Proceedings of the Fifth International Conference on Networking and Services (ICNS'09)*. 2009. IEEE.

72. Kellner, A., K. Behrends, and D. Hogrefe, *Simulation environments for wireless sensor networks*, in *Institute of Computer Science*. 2010, Georg-August-Universität Göttingen: Germany. p. 13.

73. Moravek, P., D. Komosny, and M. Simek, *Specifics of WSN simulations.* Elektrorevue, 2011. 2(3): p. 34-40.

74. Stehlık, M., *Comparison of simulators for wireless sensor networks*, in *Faculty of Informatics*. 2011, Masaryk University: Brno, Czech Republic. p. 87.

75. Sundani, H., et al., *Wireless sensor network simulators a survey and comparisons.* International Journal of Computer Networks, 2011. 2(5): p. 249-265.

76. Paul, D.C., *A computational investigation of wireless sensor network simulation*, in *Proceedings of the 50th Annual Southeast Regional Conference*. 2012, ACM. p. 401-402.

77. Kumar, A., et al. *Simulators for wireless networks: A comparative study*. in *International Conference on Computing Sciences (ICCS)*. 2012. IEEE.

78. Abuarqoub, A., et al. *Simulation issues in wireless sensor networks: A survey*. in *The Sixth International Conference on Sensor Technologies and Applications (SENSORCOMM 2012)*. 2012.

79. Kumar, R. and S. Goyal, *Perspective of WSNs simulators.* International Journal of Information Engineering, 2013. 3(2): p. 37-44.

80. Lahmar, K., R. Chéour, and M. Abid. *Wireless sensor networks: Trends, power consumption and simulators*. in *Modelling Symposium (AMS), 2012 Sixth Asia*. 2012. IEEE.

81. Ali, Q.I., *Simulation Framework of Wireless Sensor Network (WSN) Using MATLAB/SIMULINK Software.* MATLAB–A Fundamental Tool for Scientific Computing and Engineering Applications, 2012. 2.

82. Cheour, R., et al. *Choice of efficient simulator tool for wireless sensor networks*. in *Proceedings (M&N), 2013 IEEE International Workshop on Measurements and Networking* 2013. IEEE.

83. Mieyeville, F., et al., *energy-Centric Simulation and Design Space exploration for Wireless Sensor networks*, in *Wireless Sensor Networks: Current Status and Future Trends*. 2012. p. 215-252.

84. Chhimwal, P., D.S. Rai, and D. Rawat, *Comparison Between Different Wireless Sensor Simulation Tools.* IOSR Journal of Electronics and Communication Engineering, 2013. 5(2): p. 54-60.

85. Gupta, S.G., et al., *Open-Source Network Simulation Tools: An Overview*. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 2013. 2(4): p. 1629-1635.

86. Chand, B.S., K.R. Rao, and S.S. Babu, *Exploration of New Simulation Tools for Wireless Sensor Networks.* International Journal of Science and Research (IJSR), 2013. 2(4): p. 269-273.

87. Sethi, A., J. Saini, and M. Bisht, *Wireless adhoc network simulators: Analysis of characterstic features, scalability, effectiveness and limitations.* International Journal of Applied Information Systems (IJAIS), 2012. 5(9).

88. Chandrasekaran, V., S. Anitha, and A. Shanmugam, *A Research Survey on Experimental Tools for Simulating Wireless Sensor Networks.* International Journal of Computer Applications, 2013. 79(16): p. 1-9.

89. Khalifa, A., et al. *Internationalized approach to wireless networks simulation*. in *21st Telecommunications Forum (TELFOR)*. 2013. IEEE.

90. Bhatt, N. and D. Kathiriya, *Comarison and Analysis of Simulators for Ad hoc Wireless Networks.* IOSR Journal of Engineering, 2013. 3(12): p. 14-22.

91. ZVKOVIC, M., et al., *A survey and classification of wireless sensor networks simulators based on the domain of use.* Ad-hoc & sensor wireless networks, 2014. 20(3-4): p. 245-287.

92. Sahin, D. and H.M. Ammari, *Programming Languages, Network Simulators, and Tools*, in *The Art of Wireless Sensor Networks*, H.M. Ammari, Editor. 2014, Springer. p. 739-788.

93. Dhviya, V. and R. Arthi, *Analysis of Simulation Tools for Underwater Wireless Sensor Networks.* International Journal of Computer Science & Engineering Technology, 2014. 5(10): p. 952-958.

94. Roy, A. and A.K. Jain, *A Survey of Wireless Network Simulators.* Journal of Multimedia Technology & Recent Advancements

2015. 2(1): p. 12–16.

95. Minakov, I., et al., *A Comparative Study of Recent Wireless Sensor Network Simulators.* ACM Transactions on Sensor Networks (TOSN), 2016. 12(3): p. 1-39.

96. Fahmy, H.M.A., *Simulators and Emulators for WSNs*, in *Wireless Sensor Networks*. 2016, Springer. p. 381-491.

97. *Ptolemy II* 14 January 2017]; Available from: http://ptolemy.eecs.berkeley.edu/ptolemyII/.

98. Cheong, E., E.A. Lee, and Y. Zhao. *Viptos: a graphical development and simulation environment for tinyos-based wireless sensor networks*. in *SenSys*. 2005.

99. *Viptos*. 14 January 2017]; Available from: http://ptolemy.berkeley.edu/viptos/.

100. Baldwin, P., et al., *Visualsense: Visual modeling for wireless and sensor network systems*. 2005, Electronics Research Laboratory, College of Engineering, University of California.

101. *VisualSense*. 14 January 2017]; Available from: http://ptolemy.berkeley.edu/visualsense/.

102. Downard, I.T., *Simulating sensor networks in ns-2*. 2004, DTIC Document.

103. *NS-2*. 14 January 2017]; Available from: http://www.isi.edu/nsnam/ns/.

104. *Mannasim*. 14 January 2017]; Available from: http://www.mannasim.dcc.ufmg.br/.

105. *NRL Sensorsim*. 14 January 2017]; Available from: http://www.nrl.navy.mil/itd/ncs/products/sensorsim.

106. Pagano, P., M. Chitnis, and G. Lipari. *RTNS: an NS-2 extension to simulate wireless real-time distributed systems for structured topologies*. in *Proceedings of the 3rd International Conference on Wireless Internet*. 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

107. Pagano, P., et al., *Simulating real-time aspects of wireless sensor networks.* EURASIP Journal on Wireless Communications and Networking, 2010. 2010: p. 19.

108. *RTNS*. 14 January 2017]; Available from: http://rtns.sssup.it/RTNSWebSite/RTNS.html.

109. Chatzigiannakis, I., et al. *TRAILS, a toolkit for efficient, realistic and evolving models of mobility, faults and obstacles in wireless networks*. in *Proceedings of the 41st Annual Simulation Symposium (ANSS)*. 2008. IEEE.

110. *PiccSIM*. 14 January 2017]; Available from: http://wsn.aalto.fi/en/tools/piccsim/.

111. Henderson, T.R., et al. *ns3-Project Goals*. in *Proceeding from the 2006 Workshop on NS-2: the IP Network Simulator*. 2006.

112. *NS-3*. 14 January 2017]; Available from: http://www.nsnam.org/.

113. Riliskis, L. and E. Osipov, *Symphony: A Framework for Accurate and Holistic WSN Simulation.* Sensors, 2015. 15(3): p. 4677-4699.

114. *Symphony*. 14 January 2017]; Available from: http://bitbucket.org/Northshoot/symphony/overview.

115. Mallanda, C., et al., *Simulating wireless sensor networks with omnet++*. IEEE Computer, 2005.

116. *OMNet++*. 14 January 2017]; Available from: http://omnetpp.org/.

117. Varga, A. and R. Hornig. *An overview of the OMNeT++ simulation environment*. in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

118. Mallanda, C.D., *Sensorsimulator: Simulation framework for sensor networks*, in *Computer Science Department*. 2005, Louisiana State University p. 49.

119. *SENSIM*. 14 January 2017]; Available from: http://csc.lsu.edu/~iyengar/publications_sensor.html#papers.

120. Suri, A., *Simulation Study for Wireless Sensor Networks and Load Sharing Routing Protocol to Increase Network Life and Connectivity*. 2005, Citeseer.

121. Pediaditakis, D., S. Mohajerani, and A. Boulis. *Poster abstract: castalia: the difference of accurate simulation in WSN*. in *Proceedings of the 4th European Conference on Wireless Sensor Networks*. 2007.

122. *Castalia*. 14 January 2017]; Available from: http://castalia.forge.nicta.com.au/index.php/en/.

123. Yi, J.M., M.J. Kang, and D.K. Noh. *SolarCastalia—Solar energy harvesting wireless sensor network simulator*. in *Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC)*. 2014. IEEE.

124. *MiXiM*. 14 January 2017]; Available from: http://mixim.sourceforge.net.

125. *NesCT*. 14 January 2017]; Available from: http://nesct.sourceforge.net/.

126. Weber, D., J. Glaser, and S. Mahlknecht. *Discrete event simulation framework for power aware wireless sensor networks*. in *Proceedings of the 5th IEEE International Conference on Industrial Informatics*. 2007. IEEE.

127. *PAWiS*. 14 January 2017]; Available from: http://pawis.sourceforge.net/.

128. Zeng, X., R. Bagrodia, and M. Gerla. *GloMoSim: a library for parallel simulation of large-scale wireless networks*. in *Proceedings of the Twelfth Workshop on Parallel and Distributed Simulation (PADS)*. 1998. IEEE.

129. *GloMoSim*. 14 January 2017]; Available from: http://pcl.cs.ucla.edu/projects/glomosim/.

130. Siraj, S., A. Gupta, and R. Badgujar, *Network simulation tools survey*. International Journal of Advanced Research in Computer and Communication Engineering, 2012. 1(4): p. 199-206.

131. Varshney, M., et al. *squalnet: An accurate and scalable evaluation framework for sensor networks*. in *Information Processing in Sensor Networks*. 2007.

132. *QualNet*. 14 January 2017]; Available from: http://web.scalable-networks.com/content/QualNet.

133. Vasu, B., et al. *Squalnet: a scalable simulation framework for sensor networks*. in *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*. 2005. ACM.

134. Varshney, M., et al. *SenQ: a scalable simulation and emulation environment for sensor networks*. in *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*. 2007. ACM.

135. Fraboulet, A., G. Chelius, and E. Fleury. *Worldsens: development and prototyping tools for application specific wireless sensors networks*. in *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN 2007)* 2007. IEEE.

136. *Worldsens*. 14 January 2017]; Available from: http://www.iot-lab.info/.

137. *WSNet*. 14 January 2017]; Available from: http://wsnet.gforge.inria.fr/.

138. *AlgoSenSim*. 14 January 2017]; Available from: http://tcs.unige.ch/doku.php/code/algosensim/overview.

139. *NetTopo*. 14 January 2017]; Available from: http://sourceforge.net/projects/nettopo/.

140. Chen, G., et al., *SENSE: a wireless sensor network simulator*, in *Advances in pervasive computing and networking*, B.K. Szymanski and B. Yener, Editors. 2005, Springer. p. 249-267.

141. *SENSE*. 14 January 2017]; Available from: http://www.ita.cs.rpi.edu/.

142. Barr, R., Z.J. Haas, and R. Van Renesse. *Jist: Embedding simulation time into a virtual machine*. in *EuroSim congress on modelling and simulation*. 2004.

143. *JiST/SWANS*. 14 January 2017]; Available from: http://jist.ece.cornell.edu/.

144. *Sinalgo*. 14 January 2017]; Available from: http://disco.ethz.ch/projects/sinalgo/.

145. *SimPy*. 14 January 2017]; Available from: http://simpy.readthedocs.org/en/latest/.

146. Eriksson, J., et al. *Mspsim–an extensible simulator for msp430-equipped sensor boards*. in *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*. 2007.

147. *MSPSim*. 14 January 2017]; Available from: https://github.com/mspsim/mspsim.

148. Osterlind, F., et al. *Cross-level sensor network simulation with cooja*. in *Proceedings of the 31st IEEE Conference on Local Computer Networks*. 2006. IEEE.

149. *COOJA*. 14 January 2017]; Available from: http://www.contiki-os.org/.

150. Sobeih, A., et al., *J-Sim: a simulation and emulation environment for wireless sensor networks.* Wireless Communications, IEEE, 2006. 13(4): p. 104-119.

151. *J-Sim*. 14 January 2017]; Available from: https://sites.google.com/site/jsimofficial/.

152. Neves, P.A., I.D. Veiga, and J.J. Rodrigues. *G-JSIM—a GUI tool for Wireless Sensor Networks simulations under J-SIM*. in *International Symposium on Consumer Electronics (ISCE)*. 2008. IEEE.

153. McGrath, D., et al. *NetSim: a distributed network simulation to support cyber exercises*. in *Proceedings of The Huntsville Simulation Conference* 2004.

154. *NetSim*. 14 January 2017]; Available from: http://tetcos.com/netsim_gen.html.

155. *OPNET*. 14 January 2017]; Available from: http://www.riverbed.com/products/performance-management-control/opnet.html.

156. *OPNET IT Guru*. 14 January 2017]; Available from: http://www.OPNET.com/university_program/itguru_academic_edition/.

157. *SSFNeT*. 14 January 2017]; Available from: http://www.ssfnet.org/homePage.html.

158. Wang, S., C. Chou, and C. Lin, *The design and implementation of the NCTUns network simulation engine.* Simulation Modelling Practice and Theory, 2007. 15(1): p. 57-81.

159. *NCTUns*. 14 January 2017]; Available from: http://nsl.cs.nctu.edu.tw/NSL/nctuns.html.

160. Wang, S.-Y. and C.-C. Lin. *NCTUns 6.0: a simulator for advanced wireless vehicular network research*. in *IEEE 71st Vehicular Technology Conference (VTC 2010-Spring)*. 2010. IEEE.

161. Fummi, F., et al., *SystemC co-simulation for core-based embedded systems.* Design Automation for Embedded Systems, 2007. 11(2-3): p. 141-166.

162. *SystemC*. 14 January 2017]; Available from: http://github.com/systemc/systemc-2.3.

163. Orebaugh, A., G. Ramirez, and J. Beale, *Wireshark & Ethereal network protocol analyzer toolkit*. 2006, Rockland, MA: Syngress.

164. *Wireshark*. 14 January 2017]; Available from: http://www.wireshark.org/.

165. *MATLAB SIMULINK*. 14 January 2017]; Available from: http://www.mathworks.com/.

166. Ali, Q.I., A. Abdulmaowjod, and H.M. Mohammed. *Simulation & performance study of wireless sensor network (WSN) using MATLAB*. in *Proceedings of the 1st International Conference on Energy, Power and Control (EPC-IQ)*. 2010. IEEE.

167. Harding, C., A. Griffiths, and H. Yu. *An Interface between MATLAB and OPNET to Allow Simulation of WNCS with MANETs*. in *International Conference on Networking, Sensing and Control*. 2007. IEEE.

168. *LabVIEW*. 14 January 2017]; Available from: http://www.ni.com/labview/.

169. Wightman, P.M. and M.A. Labrador. *Atarraya: A simulation tool to teach and research topology control algorithms for wireless sensor networks*. in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

170. *Atarraya*. 18 January 2017]; Available from: http://www.cse.usf.edu/~labrador/Atarraya/.

171. Qela, B., G. Wainer, and H. Mouftah. *Simulation of large wireless sensor networks using Cell-Devs*. in *Proceedings of the Winter Simulation Conference (WSC)*. 2009. IEEE.

172. *Cell-DEVS*. 14 January 2017]; Available from: http://cell-devs.sce.carleton.ca/mediawiki/index.php/Main_Page.

173. Noormohammadpour, M., et al., *ABMQ: An Agent-Based Modeler and Simulator for Self-Organization in MANETs using Qt.* arXiv preprint arXiv:1312.2241, 2013.

174. Zhang, B., *Performance Management for Energy Harvesting Wireless Sensor Networks*, in *Department of Computer Science*. 2012, George Mason University: Fairfax, VA. p. 118.

175. Luke, S., et al., *Mason: A multiagent simulation environment.* Simulation, 2005. 81(7): p. 517-527.

176. *MASON*. 14 January 2017]; Available from: http://cs.gmu.edu/~eclab/projects/mason/.

177. *Repast3*. 14 January 2017]; Available from: http://repast.sourceforge.net/repast_3/index.html.

178. *RepastSNS*. 14 January 2017]; Available from: http://www.iiia.csic.es/~mpujol/RepastSNS/.

179. Collier, N., *Repast: An agent based modelling toolkit for java*. 2001, Social Science Research Computing, University of Chicago: Chicago, IL.

180. del Carmen Delgado-Roman, M., M. Pujol-Gonzalez, and C. Sierra, *Multiagent Co-ordination of Wireless Sensor Networks*, in *Citizen in Sensor Networks*, J. Nin and D. Villatoro, Editors. 2013, Springer. p. 19-32.

181. *Energy Efficiency Simulation in NetLOGO*. 14 January 2017]; Available from: http://ccl.northwestern.edu/netlogo/models/community/WSN-Project-20-Final-6

182. *Data Dissemination in NetLoGO*. 14 January 2017]; Available from: http://ccl.northwestern.edu/netlogo/models/community/WSN.

183. Haghighi, M. and D. Cliff. *Sensomax: An agent-based middleware for decentralized dynamic data-gathering in wireless sensor networks*. in *International Conference on Collaboration Technologies and Systems (CTS)*. 2013. IEEE.

184. Haghighi, M. *An agent-based multi-model tool for simulating multiple concurrent applications in wsns*. in *Journal of Advances in Computer Networks (JACN), 5th International Conference on Communication Software and Networks*. 2013.

185. Barton, J.J. and V. Vijayaraghavan, *UBIWISE, a simulator for ubiquitous computing systems design.* Hewlett-Packard Laboratories Palo Alto, â AI HPL-2003-93, 2003.

186. *UbiWise*. 14 January 2017]; Available from: http://home.comcast.net/~johnjbarton/ubicomp/ur/ubiwise/.

187. Campuzano, F., et al., *Flexible simulation of ubiquitous computing environments*, in *Ambient Intelligence-Software and Applications*. 2011, Springer. p. 189-196.

188. *UbikSim*. 14 January 2017]; Available from: http://github.com/emilioserra/UbikSim/wiki.

189. O'Neill, E., *TATUS A Ubiquitous Computing Simulator*, in *Trinity College,*. 2005, The University of Dublin: Dublin, Republic of Ireland.

190. Dhurandher, S.K., et al., *UWSim: A simulator for underwater sensor networks.* Simulation, 2008. 84(7): p. 327-338.

191. *UWSim*. 14 January 2017]; Available from: http://www.irs.uji.es/uwsim/.

192. Baldo, N., et al. *ns2-MIRACLE: a modular framework for multi-technology and cross-layer support in network simulator 2*. in *Proceedings of the 2nd International Conference on Performance Evaluation Methodologies and Tools*. 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

193. Petrioli, C. and R. Petroccia, *SUNSET: Simulation, emulation and real-life testing of underwater wireless sensor networks.* Proceedings of IEEE UComms 2012, 2012: p. 12-14.

194. Petrioli, C., R. Petroccia, and D. Spaccini. *SUNSET version 2.0: Enhanced framework for simulation, emulation and real-life testing of underwater wireless sensor networks*. in *Proceedings of the Eighth ACM International Conference on Underwater Networks and Systems*. 2013. ACM.

195. *SUNSET*. 14 January 2017]; Available from: http://reti.dsi.uniroma1.it/UWSN_Group/index.php?page=sunset&sec=tech_desc.

196. Petrioli, C., et al. *The SUNRISE GATE: accessing the SUNRISE federation of facilities to test solutions for the Internet of Underwater Things*. in *Underwater Communications and Networking (UComms)*. 2014. IEEE.

197. *SUNRISE*. 14 January 2017]; Available from: http://fp7-sunrise.eu/.

198. Masiero, R., et al. *DESERT Underwater: an NS-Miracle-based framework to DEsign, Simulate, Emulate and Realize Test-beds for Underwater network protocols*. in *OCEANS*. 2012. Yeosu: IEEE.

199. *DESERT*. 14 January 2017]; Available from: http://nautilus.dei.unipd.it/desert-underwater.

200. Toso, G., et al. *RECORDS: a remote control framework for underwater networks*. in *13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*. 2014. IEEE.

201. *RECORDS*. 14 January 2017]; Available from: http://nautilus.dei.unipd.it/desert-underwater.

202. Peng, Z., et al. *Aqua-Net: An underwater sensor network architecture: Design, implementation, and initial testing*. in *OCEANS*. 2009. Biloxi: IEEE.

203. *Aqua-Net*. 14 January 2017]; Available from: http://obinet.engr.uconn.edu/wiki/index.php?title=Aqua-Net&redirect=no.

204. Le, S.N., et al. *Sealinx: A multi-instance protocol stack architecture for underwater networking*. in *Proceedings of the Eighth ACM International Conference on Underwater Networks and Systems*. 2013. ACM.

205. *SeaLinx*. 14 January 2017]; Available from: http://www.oceantune.org/index.php/79-ocean-tune/network-solution/71-sealinx.

206. Zhu, Y., et al. *Aqua-Net Mate: A real-time virtual channel/modem simulator for Aqua-Net*. in *OCEANS*. 2013. Bergen: IEEE.

207. Peng, Z., et al. *An underwater network testbed: design, implementation and measurement*. in *Proceedings of the Second Workshop on Underwater Networks*. 2007. ACM.

208. *Aqua-Lab*. 14 January 2017]; Available from: http://obinet.engr.uconn.edu/wiki/index.php?title=Aqua-Lab&redirect=no.

209. Xie, P., et al. *Aqua-Sim: an NS-2 based simulator for underwater sensor networks*. in *OCEANS*. 2009. IEEE.

210. *Aqua-Sim*. 14 January 2017]; Available from: http://obinet.engr.uconn.edu/wiki/index.php?title=Aqua-Sim&redirect=no.

211. Peng, Z., et al. *Aqua-TUNE: A testbed for underwater networks*. in *OCEANS*. 2011. Spain: IEEE.

212. Dhurandher, S.K., M.S. Obaidat, and M. Gupta, *An acoustic communication based AQUA-GLOMO simulator for underwater networks.* Human-centric Computing and Information Sciences, 2012. 2(1): p. 1-14.

213. Sehgal, A., I. Tumar, and J. Schönwälder. *Aquatools: An underwater acoustic networking simulation toolkit*. in *OCEANS*. 2010. IEEE.

214. Noh, Y., D. Torres, and M. Gerla, *Software-defined underwater acoustic networking platform and its applications.* Ad Hoc Networks, 2015.

215. *UANT*. 14 January 2017]; Available from: http://github.com/nesl/uant.

216. Guerra, F., P. Casari, and M. Zorzi. *World Ocean Simulation System (WOSS): a simulation tool for underwater networks with realistic propagation modeling*. in *Proceedings of the Fourth ACM International Workshop on UnderWater Networks*. 2009. ACM.

217. *WOSS*. 14 January 2017]; Available from: http://telecom.dei.unipd.it/ns/woss/.

218. Wolff, L.M., E. Szczepanski, and S. Badri-Hoeher. *Acoustic underwater channel and network simulator*. in *OCEANS*. 2012. Yeosu: IEEE.

219. Phoha, S., E.M. Peluso, and R.L. Culver, *A high-fidelity ocean sampling mobile network (SAMON) simulator testbed for evaluating intelligent control of unmanned underwater vehicles*. IEEE Journal of Oceanic Engineering, 2001. 26(4): p. 646-653.

220. Ovaliadis, K. and N. Savage, *Underwater Sensor Network Simulation Tool (USNeT).* International Journal of Computer Applications, 2013. 71(22): p. 19-27.

221. *Prowler*. 14 January 2017]; Available from: http://www.isis.vanderbilt.edu/projects/nest/prowler/.

222. *Wireless Sensor Network Localization Simulator*. 14 January 2017]; Available from: http://www.codeproject.com/Articles/606364/Wireless-Sensor-Network-Localization-Simulator-v.

223. *Sensor Security Simulator (S3)*. 14 January 2017]; Available from: http://www.fi.muni.cz/~xsvenda/s3.html.

224. Kröller, A., et al., *Shawn: A new approach to simulating wireless sensor networks*. 2005, Proceedings Design, Analysis, and Simulation of Distributed Systems (DASD05). p. 117-124.

225. *Shawn*. 14 January 2017]; Available from: http://github.com/itm/shawn/.

226. Ghica, O., *SIDnet-SWANS manual*. 2010.

227. *SIDnet-SWANS* 14 January 2017]; Available from: http://users.eecs.northwestern.edu/~ocg474/SIDnet.html.

228. Mármol, F.G. and G.M. Pérez. *TRMSim-WSN, trust and reputation models simulator for wireless sensor networks*. in *International Conference on Communications (ICC'09)*. 2009. IEEE.

229. *TRMSim-WSN* 14 January 2017]; Available from: http://ants.inf.um.es/~felixgm/research/trmsim-wsn/.

230. Marchiori, A., et al. *Realistic performance analysis of wsn protocols through trace based simulation*. in *Proceedings of the 7th ACM Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*. 2010. ACM.

231. Sundresh, S., W. Kim, and G. Agha. *SENS: A sensor, environment and network simulator*. in *Proceedings of the 37th Annual Symposium on Simulation*. 2004. IEEE Computer Society.

232. *SENS*. 14 January 2017]; Available from: http://osl.cs.illinois.edu/sens/.

233. Ben-Asher, Y., et al., *IFAS: Interactive flexible ad hoc simulator.* Simulation Modelling Practice and Theory, 2007. 15(7): p. 817-830.

234. Mount, S., et al., *Sensor: an algorithmic simulator for wireless sensor networks.* In Proceedings of Eurosensors, 2006. 20: p. 400-411.

235. *Dingo*. 14 January 2017]; Available from: http://code.google.com/p/dingo-wsn/.

236. Kelly IV, C., V. Ekanayake, and R. Manohar. *SNAP: A sensor-network asynchronous processor*. in *Proceedings of the Ninth International Symposium on Asynchronous Circuits and Systems*. 2003. IEEE.

237. *SNAP*. 14 January 2017]; Available from: http://vlsi.cornell.edu/~rajit/ps/snap.ps.gz.

238. Ould-Ahmed-Vall, E., et al. *Simulation of large-scale sensor networks using GTSNetS*. in *Proceeding of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*. 2005. IEEE.

239. *GTSNetS*. 14 January 2017]; Available from: http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/index.html.

240. Du, W., F. Mieyeville, and D. Navarro. *IDEA1: A SystemC-based system-level simulator for wireless sensor networks*. in *International Conference on Wireless Communications, Networking and Information Security (WCNIS)*. 2010. IEEE.

241. *IDEA1*. 14 January 2017]; Available from: http://idea1inl.free.fr/IDEA1/.

242. *WiseNet*. 14 January 2017]; Available from: http://code.google.com/p/secwsnsim/.

243. Wen, Y., et al. *SimGate: Full-System, Cycle-Close Simulation of the Stargate Sensor Network Intermediate Node*. in *Proceedings of International Conference on Embedded Computer Systems: Architectures, MOdeling, and Simulation (IC-SAMOS)*. 2006.

244. Xu, C., et al., *Simsync: A time synchronization simulator for sensor networks.* Acta Automatica Sinica, 2006. 32(6): p. 1008-1014.

245. Yi, S., et al., *SensorMaker: A wireless sensor network simulator for scalable and fine-grained instrumentation*, in *Computational Science and Its Applications–ICCSA 2008*, O. Gervasi, et al., Editors. 2008, Springer. p. 800-810.

246. Barbancho, J., et al., *OLIMPO, An Ad-Hoc Wireless Sensor Network Simulator for Optimal SCADA-Applications.* Communication Systems and Networks (CSN 2004), 2004. 450.

247. Wen, Y., R. Wolski, and G. Moore. *Disens: scalable distributed sensor network simulation*. in *Proceedings of the 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. 2007. ACM.

248. Wen, Y., R. Wolski, and S. Gurun. *S 2 DB: a novel simulation-based debugger for sensor network applications*. in *Proceedings of the 6th ACM & IEEE International Conference on Embedded Software*. 2006. ACM.

249. Lim, H.B., et al. *WISDOM: simulation framework for middleware services in wireless sensor networks*. in *Proceedings of the 5th IEEE Consumer Communications and Networking Conference (CCNC 2008)* 2008. IEEE.

250. Al-Karaki, J.N. and G. Al-Mashaqbeh. *SENSORIA: A new simulation platform for wireless sensor networks*. in *International Conference on Sensor Technologies and Applications (SensorComm 2007)*. 2007. IEEE.

251. Sha, K., Z. Zhu, and W. Shi, *Capricorn: A large scale wireless sensor network simulator*. 2004, Wayne State University: Detroit, MI.

252. Mochocki, B.C. and G.R. Madey. *H-MAS: a heterogeneous, mobile, ad-hoc sensor-network simulation environment*. in *Proceedings of the 7th Annual Swarm Users/Researchers Conference*. 2003.

253. Li, C.-W., et al., *SNSim: Study and Implementation of a Wireless Sensor Network Simulator*. Journal of Chinese Computer Systems, 2010. 31(6): p. 1025-1029.

254. Sinha, S., Z. Chaczko, and R. Klempous, *SNIPER: A Wireless Sensor Network Simulator*, in *Computer Aided Systems Theory-EUROCAST 2009*. 2009, Springer. p. 913-920.

255. Boulis, A., et al. *CaVi--Simulation and Model Checking for Wireless Sensor Networks*. in *Proceedings of the Fifth International Conference on Quantitative (QEST'08)* 2008. IEEE.

256. Karagiannis, M., I. Chatzigiannakis, and J. Rolim. *WSNGE: A platform for simulating complex wireless sensor networks supporting rich network visualization and online interactivity*. in *Proceedings of the 2009 Spring Simulation Multiconference*. 2009. Society for Computer Simulation International.

257. Menichelli, F. and M. Olivieri, *TikTak: A Scalable Simulator of Wireless Sensor Networks Including Hardware/Software Interaction.* Wireless Sensor Network, 2010. 2(11): p. 815-822.

258. *ShoX*. 14 January 2017]; Available from: http://shox.sourceforge.net/.

259. *PASENS*. 14 January 2017]; Available from: http://user.chol.com/~legnamai/pasens/.

260. Samper, L., et al. *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*. 2006. ACM.

261. *Glonemo*. 14 January 2017]; Available from: http://rml.lri.fr/glonemo/.

262. Riliskis, L. and E. Osipov, *Maestro: An Orchestration Framework for Large-Scale WSN Simulations.* Sensors, 2014. 14(3): p. 5392-5414.

263. Mehdi, K., et al. *CupCarbon: a multi-agent and discrete event wireless sensor network design and simulation tool*. in *Proceedings of the 7th International ICST Conference on Simulation Tools and Techniques*. 2014. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

264. *CupCarbon*. 14 January 2017]; Available from: http://www.cupcarbon.com/.

265. Yan, C., et al. *TimSim: A Timestep-Based Wireless Ad-Hoc Network Simulator*. in *Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 2013. Melbourne, VIC: IEEE.

266. Ribeiro, D.H., et al. *JSensor: A parallel simulator for wireless sensor networks and distributed systems*. in *Proceedings of the 41st International Conference on Parallel Processing Workshops (ICPPW)*. 2012. IEEE.

267. *JSensor*. 14 January 2017]; Available from: http://www.joubertlima.com.br/JSensor/.

268. Levis, P., et al. *TOSSIM: Accurate and scalable simulation of entire TinyOS applications*. in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*. 2003. ACM.

269. *TinyOS*. 14 January 2017]; Available from: http://www.tinyos.net.

270. Mora-Merchan, J.M., et al., *mTOSSIM: A simulator that estimates battery lifetime in wireless sensor networks.* Simulation Modelling Practice and Theory, 2013. 31: p. 39-51.

271. *PowerTOSSIM*. 14 January 2017]; Available from: http://www.eecs.harvard.edu/~shnayder/ptossim/.

272. Perla, E., et al. *PowerTOSSIM z: realistic energy modelling for wireless sensor network environments*. in *Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*. 2008. ACM.

273. *PowerTOSSIM z*. 14 January 2017]; Available from: http://www.scss.tcd.ie/disciplines/computer_systems/ccg/software/powertossimz/.

274. Perrone, L.F. and D.M. Nicol. *A scalable simulator for TinyOS applications*. in *Proceedings of the Winter Simulation Conference*. 2002. IEEE.

275. Demmer, M., et al., *Tython: a dynamic simulation environment for sensor networks*. 2005: Computer Science Division, University of California.

276. *TYTHON*. 14 January 2017]; Available from: http://www.tinyos.net/tinyos-1.x/doc/tython/tython.html.

277. Watson, D. and M. Nesterenko. *Mule: Hybrid simulator for testing and debugging wireless sensor networks*. in *Workshop on Sensor and Actor Network Protocols and Applications*. 2004.

278. Titzer, B.L., D.K. Lee, and J. Palsberg. *Avrora: Scalable sensor network simulation with precise timing*. in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*. 2005. IEEE.

279. *Avrora*. 14 January 2017]; Available from: http://compilers.cs.ucla.edu/avrora/.

280. *AvroraZ*. 14 January 2017]; Available from: http://citavroraz.sourceforge.net/.

281. Landsiedel, O., K. Wehrle, and S. Götz. *Accurate prediction of power consumption in sensor networks*. in *Proceedings of the Second Workshop on Embedded Networked Sensors*. 2005. Citeseer.

282. Blazakis, D., et al. *ATEMU: a fine-grained sensor network simulator*. in *Proceedings of the First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (IEEE SECON'04)*. 2004. IEEE.

283. *ATEMU*. 14 January 2017]; Available from: http://www.cshcn.umd.edu/research/atemu/.

284. Park, C. and P.H. Chou. *EmPro: an environment/energy emulation and profiling platform for wireless sensor networks*. in *Proceedings of the 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks (SECON'06)*. 2006. IEEE.

285. *OCTAVEX*. 14 January 2017]; Available from: http://www.millennium.berkeley.edu/pipermail/tinyos-help/2005-September/012224.html.

286. Dall'Ora, R., et al. *SensEH: From simulation to deployment of energy harvesting wireless sensor networks*. in *Proceedings of the 39th Conference on Local Computer Networks Workshops (LCN Workshops)* 2014. IEEE.

287. Didioui, A., et al. *HarvWSNet: A co-simulation framework for energy harvesting wireless sensor networks*. in *International Conference on Computing, Networking and Communications (ICNC)*. 2013. IEEE.

288. *UbiSec&Sens*. 14 January 2017]; Available from: http://www.ist-ubisecsens.org/.

289. Clouser, T., R. Thomas, and M. Nesterenko, *Emuli: Emulated Stimuli for Wireless Sensor Network Experimentation*. 2007, Kent State University.

290. Luo, Q., et al. *MEADOWS: modeling, emulation, and analysis of data of wireless sensor networks*. in *Proceeedings of the 1st International Workshop on Data Management for Sensor Networks: in conjunction with VLDB 2004*. 2004. ACM.

291. Maret, T., et al., *Freemote emulator: a lightweight and visual java emulator for wsn*, in *Wired/Wireless Internet Communications*, J. Harju, et al., Editors. 2008, Springer. p. 92-103.

292. *Freemote Emulator* 14 January 2017]; Available from: http://www.assembla.com/wiki/show/freemote.

293. Wu, H., et al., *VMNet: Realistic emulation of wireless sensor networks.* IEEE Transactions on Parallel and Distributed Systems, 2007. 18(2): p. 277-288.

294. *VMNeT*. 14 January 2017]; Available from: http://www.cse.ust.hk/vmnet/.

295. *WSim*. 14 January 2017]; Available from: http://wsim.gforge.inria.fr/.

296. Girod, L., et al., *Emstar: A software environment for developing and deploying heterogeneous sensor-actuator networks.* ACM Transactions on Sensor Networks (TOSN), 2007. 3(3): p. 35.

297. *EmStar*. 14 January 2017]; Available from: http://cens.ucla.edu/projects/2007/Systems/EmStar/.

298. Youssef, S.M., M.A. El-Nasr, and M. Aslan. *WiEmu: The Design and Implementation of a Flexible Agent-Based Scalable Network Emulator for Wireless Sensor Networks*. in *IACSIT*. 2012. Hong Kong: IACSIT Press, Singapore.

299. *WiEmu*. 14 January 2017]; Available from: http://wiemu.sourceforge.net/apidocs/.

300. Khanda, R., et al. *An Emulation Framework for Wireless Structural Health Monitoring*. in *Earth and Space 2010*. 2010. ASCE.

301. Zhang, J., et al. *A software-hardware emulator for sensor networks*. in *Proceedings of the 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*. 2011. Salt Lake City, UT: IEEE.

302. *SUNSHINE*. 14 January 2017]; Available from: http://rijndael.ece.vt.edu/sunshine/index.html.

303. *CORE*. 14 January 2017]; Available from: http://www.nrl.navy.mil/itd/ncs/products/core.

304. Khan, A., S. Bilal, and M. Othman, *A Performance Comparison of Network Simulators for Wireless Networks.* ICCSCE 2013.

305. Timm-Giel, A., et al., *Comparative simulations of WSN*. ICT-MobileSummit, 2008.

306. Cavin, D., Y. Sasson, and A. Schiper. *On the accuracy of MANET simulators*. in *Proceedings of the Second ACM International Workshop on Principles of Mobile Computing*. 2002. ACM.

307. Khan, M.A., H. Hasbullah, and B. Nazir. *Recent open source wireless sensor network supporting simulators: A performance comparison*. in *International Conference on Computer, Communications, and Control Technology (I4CT)*. 2014. IEEE.

308. Schoch, E., et al. *Simulation of ad hoc networks: ns-2 compared to jist/swans*. in *Proceedings of the First International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 08')*. 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

309. Gamess, E., I. Mahgoub, and M. Rathod. *Scalability evaluation of two network simulation tools for Vehicular Ad hoc Networks*. in *Wireless Advanced (WiAd)*. 2012. IEEE.

310. Xian, X., W. Shi, and H. Huang. *Comparison of OMNET++ and other simulator for WSN simulation*. in *Proceedings of the 3rd IEEE Conference on Industrial Electronics and Applications (ICIEA)* 2008. IEEE.

311. Alizai, M.H., O. Landsiedel, and K. Wehrle, *Modeling Execution Time and Energy Consumption in Sensor Node Simulation.* PIK-Praxis der Informationsverarbeitung und Kommunikation, 2009. 32(2): p. 127-132.

312. Keshteh Gar, E. and A. Sadeghi-Niaraki, *Ubi-Asthma: Design and Implementation of Asthmatic Patient Monitoring System in Ubiquitous Geospatial Information System.* Journal of Geomatics Science and Technology, 2015. 5(2): p. 55-66.

313. Sahelgozin, M., A. Sadeghi-Niaraki, and S. Dareshiri, *Proposing a Multi-Criteria Path Optimization Method in Order to Provide a Ubiquitous Pedestrian Wayfinding Service.* The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 2015. 40(1): p. 639.

314. Nikparvar, B., A. Sadeghi-Niaraki, and P. Azari, *Ubiquitous Indoor Geolocation: a Case Study of Jewellery Management System.* The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 2014. 40(2): p. 215.

315. Jamali, B. and A. Sadeghi-Niaraki, *Improving Geo-labeling in Ubiquitous Environment Based on Augmented Reality*. Journal of Geomatics Science and Technology, 2016. 5(3): p. 99-110.

316. Moosavi, S. and A. Sadeghi-Niaraki, *A Survey of Smart Electrical Boards in Ubiquitous Sensor Networks for Geomatics Applications.* The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 2015. 40(1): p. 503.

317. Islam, M., S.Q. Jalil, and M.H. Rehmani, *Role of Wireless Sensor Networks in Emerging Communication Technologies: A Review*, in *Emerging Communication Technologies Based on Wireless Sensor Networks: Current Research and Future Applications*, M.H. Rehmani and A.-S.K. Pathan, Editors. 2016, CRC Press. p. 376.

318. Akan, O.B., O.B. Karli, and O. Ergul, *Cognitive radio sensor networks.* IEEE Network, 2009. 23(4): p. 34-40.

319. Ahmad, A., et al., *A Survey on Radio Resource Allocation in Cognitive Radio Sensor Networks.* IEEE Communications Surveys & Tutorials, 2015. 17(2): p. 888-917.

320. Bukhari, S.H.R., M.H. Rehmani, and S. Siraj, *A Survey of Channel Bonding for Wireless Networks and Guidelines of Channel Bonding for Futuristic Cognitive Radio Sensor Networks.* IEEE Communications Surveys & Tutorials, 2016. 18(2): p. 924-948.

321. Shah, G.A. and O.B. Akan, *Cognitive Adaptive Medium Access Control in Cognitive Radio Sensor Networks.* IEEE Transactions on Vehicular Technology, 2015. 64(2): p. 757-767.

322. Bukhari, S.H.R., S. Siraj, and M.H. Rehmani, *NS-2 based simulation framework for cognitive radio sensor networks.* Wireless Networks, 2016: p. 1-17.

323. Niyato, D., E. Hossain, and A. Fallahi, *Sleep and Wakeup Strategies in Solar-Powered Wireless Sensor/Mesh Networks: Performance Analysis and Optimization.* IEEE Transactions on Mobile Computing, 2007. 6(2): p. 221-236.

324. Akhtar, F. and M.H. Rehmani, *Energy replenishment using renewable and traditional energy resources for sustainable wireless sensor networks: A review.* Renewable and Sustainable Energy Reviews, 2015. 45: p. 769-784.

325. Hasenfratz, D., et al. *Analysis, Comparison, and Optimization of Routing Protocols for Energy Harvesting Wireless Sensor Networks*. in *International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*. 2010.

326. Niyato, D., et al., *Wireless sensor networks with energy harvesting technologies: a game-theoretic approach to optimal energy management.* IEEE Wireless Communications, 2007. 14(4): p. 90-96.

327. Sharif, M. and A.A. Alesheikh, *Context-awareness in similarity measures and pattern discoveries of trajectories: a context-based dynamic time warping method.* GIScience & Remote Sensing, 2017: p. 1-27.

328. Alcaraz, C., et al. *Wireless sensor networks and the internet of things: Do we need a complete integration?* in *1st International Workshop on the Security of the Internet of Things (SecIoT'10)*. 2010.

329. Kurschl, W. and W. Beer, *Combining cloud computing and wireless sensor networks*, in *Proceedings of the 11th International Conference on Information Integration and Web-based Applications &amp; Services*. 2009, ACM: Kuala Lumpur, Malaysia. p. 512-518.

330. Ahmed, K. and M. Gregory. *Integrating Wireless Sensor Networks with Cloud Computing*. in *Seventh International Conference on Mobile Ad-hoc and Sensor Networks*. 2011.