DESIGN ISSUES ON SOFTWARE ASPECTS AND SIMULATION TOOLS FOR WIRELESS SENSOR NETWORKS

Leelavathi G[1], Shaila K[2], Venugopal K R[3] and L M Patnaik[4]

[1]Department of Electronics and Communication, Govt.S.K.S.J.Technological Institute, Bangalore, India
[2]Professor and Head, Department of Electronics and Communication, Vivekananda Institute of Technology, Bangalore, India
[3]Principal, University Visvesvaraya College of Engineering, Bangalore, India
[4]Honorary Professor, Indian Institute of Science, Bangalore, India
leelavathig12@yahoo.in

ABSTRACT

In this paper, various existing simulation environments for general purpose and specific purpose WSNs are discussed. The features of number of different sensor network simulators and operating systems are compared. We have presented an overview of the most commonly used operating systems that can be used in different approaches to address the common problems of WSNs. For different simulation environments there are different layer, components and protocols implemented so that it is difficult to compare them. When same protocol is simulated using two different simulators still each protocol implementation differs, since their functionality is exactly not the same. Selection of simulator is purely based on the application, since each simulator has a varied range of performance depending on application.

KEYWORDS

Emulator, Simulator, Operation System, Programming Languages, Wireless Sensor Network.

1 INTRODUCTION

1.1 Wireless Sensor Networks

Wireless Sensor Network [1][2][3][4] is a collection of tiny nodes. Wireless Sensor Networks (WSNs) is one of the developing areas in network research communities. Each node comprises of processing units (one or more microcontrollers, CPUs or DSP chips), memories (program, data and flash memories), RF transceiver (usually with a single omni- directional antenna), power source (e.g., batteries and solar cells) and various types of sensors and actuators. These tiny sensor nodes are deployed in spatially distributed terrain. Since it is an infrastructure less network the nodes communicate wirelessly and self-organize themselves. Sensor nodes are often deployed in an ad hoc manner. The nodes sense the environment and communicate the information gathered from the monitoring field (e.g., an area) through wireless links. The information is forwarded via multiple hops, to a sink which can be a controller or a monitor, that uses it locally or the Internet through a gateway. The nodes can be either stationary or moving, may or may not be aware of their location and can be either homogeneous or heterogeneous. Sensors can be deployed in continuous changing environments or in environments that are inaccessible for humans. Sensor networks offer good solutions to many applications like health monitoring, fault

DOI : 10.5121/ijnsa.2013.5204

diagnosis and innovative human-machine interaction paradigms and spans from military applications to almost every field in day to day life.

During the design of Wireless Sensor Networks one has to consider the effects of energy consumption, fault tolerance, quality of service, synchronization, scheduling strategies, system topology, communication and coordination protocols. The simulation environments can be divided into two major categories: *adaptive* development and *new* development. The adaptive development for example NS-2, MATLAB etc., includes simulation environments that already exist before emerging of the idea of WSNs. These simulation environments were applied to support wireless networks and later introduced for the WSNs.

At present various simulation environments are available for the simulation of WSNs. The new simulators like TOSSIM, UWSim, AVRORA etc., [5][6][7] are created for simulating WSNs, considering sensor specific characteristics. The tools are selected based on the properties of the simulation tools, popularity of the tool among research communities, the possibility of providing extension and the active maintenance of WSNs. The available simulation environments differ significantly in their structure and features such as models and protocols. Some of the selected and widely used simulation environments for Wireless Sensor Networks
are discussed and compared in section 5.

1.2 Software Aspects

WSNs have several common aspects when compared with Wireless Ad-hoc Network. A Sensor Network operating system must address the characteristics of sensor network that is necessary for developing and maintaining the WSNs. For example, the ability to dynamically reprogram WSNs. This paper deals with the aspects focusing on technologies and standards along with the modifications and improvements in both operating systems and simulators. In WSNs, the important area of research is the simulation environment. It is observed through literature survey each research groups use different simulation environments for performance evaluation.

1.3 Simulation in WSNs

The performance analysis of wired and wireless networks is broadly classified into three groups. They are Analytical Methods, Computer Simulation and Physical Measurement. The resource constraints imposed on sensor networks, such as energy limitation, decentralized collaboration, and fault tolerance results in the use of complex algorithms for sensor networks that usually disobey analytical methods. The quantitative analysis of sensor networks indicates that simulation is the feasible approach because implementation of experiments on a test bed is costly and difficult. On the other hand, repeatability is largely compromised since many factors like routing, energy consumption, topology of the network affect the experimental results.

Real time execution of experiments is time consuming, hence simulation becomes essential for the developments of WSNs, so that protocols and schemes can be evaluated on a very large scale. WSNs simulators allow users to segregate different factors by tweaking configurable parameters like the number of nodes, computational power, coverage area etc. There are no general protocols or algorithms developed for a sensor network since the protocols have to be developed for a particular application with specific features. With the above factors discussed it is vital to develop simulation platforms that are constructive to investigate both the networking issues and the distributed computing aspects of Wireless Sensor Networks.

In the present day, simulation scenarios for Wireless Sensor Networks largely focus on the networking issues rather than exploring distributed computing issues. The research area of WSNs is a continuous process hence, new protocols and components are developed continuously. These protocols have to be tested, which results in increase in the cost for developing and deployment of the sensor nodes. An alternative to the development of real sensor nodes is to use simulation environments, which are widely practiced in other networking domains. In Wireless Sensor Networks, special characteristics are to be considered and made available for the simulation environments.

Organization: Section 2 discusses the related work and section 3 briefly introduces the Operating Systems and Programming Language. The different types of operating systems are addressed in section 4 followed by simulation in section 5. The different types of simulation tools are explained in section 6. Performance evaluation is discussed in section 8 and conclusions in section 9.

2 RELATED WORK

The majority of the survey on WSNs deals with the prologue of Operating Systems and Simulators of WSNs. WSNs design involves ample of aspects related to several issues like connectivity, access to the channel, signal processing techniques, design of energy efficient communication protocols etc. These aspects must be considered due to overabundance of WSNs features . In [5][6][7] the applications of WSNs and information regarding the wireless technologies are discussed along with the case studies. LiteOS [8][9] a UNIX-like, multithreaded software platform is used for WSNs application development. Using this operating system, the evaluation of the platform has been done experimentally by measuring the performance of common tasks.

A thread-driven approach is attractive in sensor networks that have been adopted for PDAs, laptops and servers. Shah et al., [10] investigates the practicality of implementing the popular approach of pre-emptively time-sliced multithreading on micro sensor nodes and explores how a thread-driven system could be tailored to the characteristics of WSNs. Author demonstrates that the added OS intricacy needs to support pre-emptive time-slicing which can be easily accommodated in MICA2 motes, with a kernel footprint cost of below 500 bytes, including the scheduler and network stack. A multithreaded system can be designed to sleep efficiently [10] when application threads indicate that there is no useful work to be done. It shows that multithreading and energy efficiency are not mutually exclusive .The hardware details specified illustrates the usage of MANTIS-OS.

For a particular application environment, David et al., [1][7][11][12][13] provide a wide-ranging analysis and comparisons of different popular sensor network simulators with a vision to help researchers to choose the superlative simulator available. J-Sim functionality [14] is enhanced with a Guided User Interface for Wireless Sensor Networks which dramatically increases the user-friendliness of the simulator.

The mechanisms provided in Contiki to meet real-time requirements and Quality of Service guarantees are discussed in [15]. An embedded real-time operating system, named HEROS *Hybrid Embedded Real-Time Operating System*, presented by Hai-ying Zhou et al., [16] which is configurable to sprint in different modes: *event-driven*, *multitask* or *hybrid* to acclimatize to diverse area of WSNs applications. Presently ,HEROS has been implemented and evaluated in different applications and on different platforms. Keeping an objective to design a configurable real time dedicated WSNOS, Zhou et al., [14] have premeditated this embedded real-time

operating system , which enables to become accustomed to an application to minimize resource consumption. HEROS has been ported on processors which are based on ARM7TDMI-S architecture, which includes NXP LPC21xx and Atmel AT91SAM7S series. A resilient operating system mechanism for Wireless Sensor Networks is presented by Kim et al., [17]. Through the experimental results on a common sensor node, it has been shown that the proposed mechanisms successfully protect the system from wayward applications.

3 OPERATING SYSTEMS AND PROGRAMMING LANGUAGES

Operating systems used for sensor networks is same as traditional embedded system such as eCos or uC/OS often designed with real-time properties. Operating systems for WSN nodes typically are less complex than general-purpose operating systems due to:

(i) The special requirements of sensor network applications.
(ii) Resource constraints in sensor network hardware platforms, Sensor network applications are usually not interactive like any other personal computers. So there is no necessity for incorporating the support for user interfaces.
(iii) Constraints in memory and memory mapping hardware support make mechanisms like virtual memory unnecessary or it is impossible to implement.

Programming the sensor nodes is more difficult than programming normal computer systems due to constrained resources, high dynamics and inaccessible deployment environments. New programming models like c@t (Computation at a point in space(@)Time), DCL (Distributed Compositional Language), galsC, nesC, Protothreads, SNACK, SQTL are developed to deal with the resource constrained environment of WSNs nodes.

Recently, maximum research is on developing operating system for sensor nodes such as, TinyOS and Colorado's Mantis Operating System (MOS). MOS is a layered multithreaded operating system with a layered network stack. Tiny OS is the most popular state machine based operating system for sensor networks.

4 OPERATING SYSTEM OVERVIEW

4.1 Tiny OS

TinyOS is a flexible application specific operating system specifically designed for Wireless Sensor Networks. It is an event-driven programming model and is not based on multithreading. TinyOS programs are self-possessed of event handlers and tasks with run to completion-semantics. When an external event occurs, TinyOS calls the appropriate event handler to handle the event viz., incoming data packet or a sensor sensing any particular data of interest. Event handlers can delay the tasks that are scheduled by the TinyOS kernel.

nesC is a special programming language.It is the extension of C programming language used by TinyOS system. Recognition of race conditions between tasks and event handlers are the important features of the nesC . TinyOS is an open source, BSD-licensed operating system intended for low-power wireless devices, which are used in different applications like sensor networks, ubiquitous computing, personal area networks, smart buildings and smart meters.

TinyOS 2.1.1 includes support for; (i) The epic, mulle, and shimmer2 platforms, (ii) Simple, uniform low-power networking across many protocols (iii) 6lowpan, an IPv6 networking layer within the TinyOS network (iv) Security on the CC2420 radio. The tiny sensor nodes in the

sensor networks executes event-centric, concurrent, reactive program that operates with severe memory and power constraints. These constraints drive towards the design of TinyOS. It supports complex, concurrent programs with very low memory requirements like 16KB of memory and the core OS is 400 bytes for many applications with minimum energy consumption.

4.2 Lite OS

LiteOS [18] is an operating system developed by UIUC for the sensor nodes. LiteOS model is an interactive and reliable model which provides the benefits like robustness, integrity and availability. It is capable of performing the simple operations when compared to TinyOS model. LiteOS provides the environment for Object-oriented languages resulting in reusability, modularity and extensibility. LiteOS provides a UNIX-like environment for sensor networks, network embedded devices and cyber physical systems. A thread-based run-time execution environment is provided by LiteOS for applications. LiteOS fits on memory-constrained nodes such as MicaZ. This operating system is multithreaded and comes bundled with a UNIX like file system and a C++ compiler. Through strong, modular environment, TinyOS and its extensions have extensively enhanced programmability of mote-class embedded devices. NesC and the event-based programming model introduce a learning curve for the most developers outside the sensor networks area.

The purpose of LiteOS is to significantly reduce the learning curve. The overall architecture of the LiteOS operating system is divided into three subsystems LiteShell, LiteFS and kernel.
LiteOS provides a wireless *node mounting mechanism*. in a file system called *LiteFS*, The *LiteShell* subsystem which runs on the base station PC side, provides Unix-like command line interface to sensor nodes. Therefore, it is a front-end that interacts with the user. The kernel subsystem of LiteOS takes the thread approach, but it also allows user applications to handle events using callback functions for efficiency.

LiteOS offers a number of characteristics in the sensor network operating systems. They are (i) a hierarchical file system and a wireless shell for user interaction using UNIX-like commands, (ii) kernel popup for dynamic loading and native execution of multithreaded applications and (iii) online debugging, dynamic memory and file system assisted communication stacks. Software upgradation is completed through careful partition between the kernel and user applications which are bridged through a suite of system calls.

A web service for LiteOS-based Wireless Sensor Networks is proposed [18] to remotely monitor the light, temperature, magnet and acceleration of the physical world. This proposed web service enables the users to remotely query and visualize the sensor readings and accepts parameterized queries, in view of the fact that UNIX-like shell commands and C programming language are supported by LiteOS. To address real time applications it is time critical and demand richer set of applications it is better to use LiteOS based Extended Service Oriented Architecture [18] for Wireless Sensor Networks. The current version of LiteOS is Version 2.1. This version includes (i)Fully integrated with AVR Studio with IRIS and MicaZ support and (ii) Fixed event logging and memory tracing functionality.

4.3 MANTIS OS

A new multithreaded cross-platform embedded operating system for Wireless Sensor Networks is provided by the MANTIS[8] MultimodAl system for NeTworks of In-situ wireless sensors. Sensor Networks are capable of handling complex tasks such as Compression, Aggregation and Signal Processing. Pre-emptive multithreading in the MANTIS sensor OS (MOS) enables micro

sensor nodes to interleave time-sensitive tasks. This feature addresses the bounded buffer producer-consumer problem. MOS is implemented in a lightweight RAM footprint that fits in less than 500 bytes of memory, including kernel, scheduler and network stack, thus achieving the memory efficiency.

The MOS power-efficient scheduler makes the microcontroller go to sleep mode after all active threads have called MOS sleep () function, reducing current consumption. An important key design feature of MOS provides flexibility in the form of cross-platform support and testing across PCs, PDAs and different micro sensor platforms. It supports remote management of in-situ sensors *by means of* dynamic reprogramming and remote login. With the intention to achieve cross-platform support, MOS was designed to influence the properties of C, a portable standard programming language. Ranging from PC's to PDA's MOS enables the same application code to execute on a different micro sensor platforms.

4.4 Contiki OS

Contiki is an operating system for embedded devices in Wireless Sensor Networks that provides powerful semantics to communicate with the sensor network. Contiki is an open source, developed in 2004. Contiki is a highly portable, multitasking operating system for memory-efficient networked embedded systems and Wireless Sensor Networks. Contiki has become fairly popular and is attaining a high-quality place in the WSNs community.

Contiki [15][19] has a broad range of networking features and is ported to various hardware platforms. The main features of Contiki are Protothreads which is an event-driven programming model. Protothreads includes special primitives to express blocking waits by adding statements that waits until an event has happened. Contiki comes with its own networking stack for TCP/IP, IPv6 and supports the radio links utilized in Wireless Sensor Networks. Preemptive threading is enabled on a per-process level and support for real-time applications provided through an extra timer called *rtime*.

4.5 HEROS

HEROS[20] is a smart, resource-aware, low-energy and distributed real-time micro-kernel, adopting the *actionetask-thread component-based multi-level system* architecture. By combining event-driven and multitask concepts, HEROS adopts component-based multi-level system architecture: action, thread and event. A minimal system element *action* in HEROS responds to the basic system operation, such as *read*, *write*, *schedule* etc.. A specific *task* called thread is a component that consists of a set of actions. An event is an *etask* (event task) which may be composed atleast by one thread.

In HEROS etask is similar to event in TinyOS and they adopt two-level scheduling policy: *non preemptive priority* scheduling for etasks and *preemptive priority* scheduling for threads. The scheduling scheme is predictable and deterministic with respect to the real time applications. A unique system interface and a system primitive-pair, i.e., *tuple* and *IN & OUT*, are proposed for system synchronization and communication. HEROS integrate the advantages of TinyOS and SDREAM and executes at different modes: *event-driven*, *multi-tasking* and *hybrid*. The combination of two kernels greatly extends the application range of HEROS from simple single-task to multi-task applications.

4.6 SOS

The SOS[21] operating system implements a static kernel that provides a wide range of services to user applications written in C and offers limited protection through kernel features. All nodes in a deployment are initialized with this static kernel. At run time, user applications made of one or more modules are loaded on top of the kernel for execution. Modules communicate with the kernel through a function jump table and with each other through indirect function references and messaging. A memory manager tracks the memory allocated by each user application. Applications in SOS can be dynamically added to, updated on and removed from nodes. To better support this functionality, the kernel intervenes when an application attempts to use a service provided by another application that is not available on the node.

4.6.1 An Online Reprogrammable Operating System

The design and implementation of hot-swapping capability in SOS[22] allows a module to be upgraded on the fly. The system uses the hot-swapping procedure which is an online process and in this system execution state of the old module can be properly transferred to the new module. It is also allowed to interface changes during hot-swapping. Hot-swapping of application modules and kernel modules are enabled. Finally, hot-swapping procedure is lightweight. To reduce the reprogramming cost in sensor nodes, during hot-swapping ,the job of module-linking is offloaded to the server. Hot-swapping, the system call performance is improved in SOS by caching the access results of the system call jump table. To evenly allocate memory traffic around the flash memory, the first-fit flash memory allocation scheme is substituted by a technique that relies on erase counters in SOS. The system is implemented on the Mica2 mote and maintenance of erase counters of each block is offloaded to the server.

4.7 RETOS

RETOS kernel detects detrimental attempts on system safety by applications and terminates the impenetrable application programs suitably [23]. The usefulness of the proposed mechanism is validated by experiments executed on a commercial sensor node device running the RETOS operating system. In Wireless Sensor Networks Resilient operating system mechanism depends on dual mode operation and static/dynamic code checking. This mechanism guarantees stack, data, code and hardware safety on MMU-less hardware without restriction of the standard C language.

4.8 Comparison of Operating System

There is a rapid development of operating systems for wireless sensor networks from 2000 onwards. It is expected that operating system should make available a platform for fast prototyping, testing and debugging application programs. TinyOS [18] and SOS [21] are based on events, Mantis [8] and TinyThreads [18] choose threads and Contiki [15] provides support for both. The comparison of operating systems with respect to diverse parameters is summarized in Table 1. In their first attempt, Casado et al., [15] presents ContikiSec, to implement a secure network layer for Wireless Sensor Network architecture for Contiki. ContikiSec provides three security modes and it has a configurable design. ContikiSec offers confidentiality, authentication and integration in communications under the Contiki operating system. In general, the expected features from the next generation WSN OS are (i) Power aware policies, (ii) Self organization, (iii) Easy interface to expose data, (iv) Simple way to program, update and debug, (v) Network applications, (vi) Power-aware communication protocols, (vii) Portability and (viii) Easy programming language.

5 SIMULATION IN WSNS

The performance analysis of wired and wireless networks can be done using analytical methods, physical measurement methods and computer simulation. The use of complex algorithms for sensor networks that frequently violate analytical methods are due to the constraints forced on sensor networks, like energy limitation, decentralized collaboration and fault tolerance. Hence, simulation is the primary feasible approach to the quantitative analysis of sensor networks [5][3][4][7].

Implementation of experiments on a test bed is costly and difficulty. While, repeatability is largely compromised since many factors like routing, energy consumption, topology of the network affect the experimental results and isolation of each factor is not possible. Running real experiments are always time consuming. Therefore, simulation becomes essential for the developments of WSNs, so that protocols, schemes and new ideas can be evaluated on a very large scale and allow users to isolate different factors by tuning configurable parameters.

There are no specific protocols or algorithms for a sensor network since it is designed for a particular application with specific features. Hence, it is essential to develop simulation platforms that are useful to explore both networking issues and distributed computing aspects of Wireless Sensor Networks. Network Simulation Tools are used which makes it difficult to apply for exploring distributed computing issues. The research area of WSNs results in continuously developing new protocols and components which have to be tested. This increases the cost for developing and deployment of sensor nodes. An alternative to the development of real sensor nodes is to use simulation environments that are widely practiced in other networking domains. In Wireless Sensor Networks, the situation is quite different because WSNs have special characteristics that are to be considered and have to be made available for simulation environments.

Table 1 Comparison of Operating Systems used in WSNs

OS / Features	LiteOS	TinyOS	Mantis	Contiki	SOS
Current license	GPL	BSD	BSD	BSD	Modified BSD
Website	www.liteos.net	www.tinyos.net	Mantis.cs.colorado.edu	www.sics.se/contiki	Projects.nesl.ucla.edu/public/sos-2x/doc/
Remote scriptable wireless shell	Yes(on the base PC, Unix commands supported)	No(application specific shell such as SimpleCmd exists)	No(on-the-mote shell is supported)	No(on-the-mote shell is supported)	No
Remote File system interface for networked nodes	Yes	No	No	No	No
File system	Hierarchial Unix-like	Single level(ELF, Matchbox)	No (will be available in 1.1)	Single level	No
Thread support	Yes	Partial (through Tiny Threads)	Yes	Yes(also supports proto threads)	No

Event based Programming	Yes(through callback functions)	Yes	No	Yes	Yes
Remote Debugging(e. g, Watch and breakpoints)	Yes	Yes (through clairvoyant)	Partial (through NodeMD)	No	No
Wireless reprogrammi ng	Yes (application level)	Yes (whole system image replacement)	No	Yes	Yes (module level)
Dynamic Memory	Yes	Yes	No	Yes	Yes
First Publication/re lease date	2007	2000	2003	2003	2005
Platform Support	MicaZ and AVR series MCU	Mica, Mica2, MicaZ, Telos, Tmote, XYZ, Iris (among others)	Mica2, MicaZ, Telos	Tmote, ESB, AVR series MCU, certain old companies	Mica2, MicaZ, XYZ
Simulator	Through AVRORA	TOSSIM, Power Tossim	Through AVRORA	Netsim, Cooja, MSPsim	Source level Simulator/Thr ough AVRORA
Current version	2.1	2.1.1	1.0 beta	2.6	No active development

A choice of operating systems are analysed with respect to different parameters and summarized in Table 1. Reprogamming and dynamic memory features are of high-quality with respect to operating systems except Mantis OS. The platform support provided by operating systems for different sensor nodes are discussed in Table 1 and provides an insight in choosing the best suitable operating system that fits for specific application.

5.1 Simulator

Simulator is a tool that is used universally to develop and test protocols of WSNs. The cost of simulating thousands of nodes in a network is very less and the simulation can be completed within very short span of the execution time. General and Specialized simulators are available to simulate WSNs.

5.2 Emulator

Emulator is a tool, which comprises of firmware as well as hardware to perform the simulation. Emulator is implemented on real nodes, providing more precision performance. Usually emulator has high scalability, which can emulate numerous sensor nodes at the same time. Emulators are different from simulators, the latter runs actual application code. Routing protocols, topology and data aggregation effects are best analyzed using simulation. Emulators are efficient for timing interactions among sensor nodes and for fine tuning network level and sensor algorithms.

6 SIMULATION TOOLS

Network simulation tools are used in the network design phase before actual implementation. A real world environment is accurately modelled and predicted the behaviour using any simulation

tool. So, that the information is provided with reference to feasibility and reflectivity that are crucial in the implementation of any systems prior to investing large amount of time and cost. It is essential to know the different simulation tools currently available and the benefits and drawbacks allied with it, so that any protocol can be built effectively with the help of simulation. This situation is, in particular true with respect to the investment in sensor networks, where hardware may have to be purchased in large quantities and at high cost. Even if the sensor nodes are readily available, testing the network for particular environment is time-consuming and difficult task. Simulation-based testing helps anyone to decide the real world environment with ease of implementation and practicality of testing large-scale networks. Therefore, simulation is the most widespread approach to develop and test new protocol for sensor networks. There are three types of simulation: *Monte Carlo Simulation*, *Trace-Driven Simulation* and *Discrete-Event Simulations*. The Trace-Driven Simulation and Discrete-Event Simulation are the two simulations which are used universally in WSNs.

6.1 Monte Carlo Simulation

To be aware of the impact of risk and uncertainty in financial, project management, cost and other forecasting models , Monte Carlo simulation [24] or probability simulation technique is used. This is a computerized mathematical technique that allows people to give an explanation for risk in quantitative analysis and decision making. This technique is used by professionals in the field of finance, project management, manufacturing, engineering, research and development, insurance, oil and gas, transportation.

Monte Carlo simulation furnishes the decision-maker with a range of promising outcomes and the probabilities of occurrence for any option of action. Monte Carlo simulation, also called probability simulation is a technique used to understand the impact of risk and uncertainty in the applications such as financial, project management, cost and other forecasting models. *Monte Carlo (MC) Method* is a computational method that utilizes random numbers. The two major applications of the MC method are (i) Multidimensional integrations (e.g., statistical mechanics in physics) and (ii) Simulation of stochastic natural phenomena (e.g., stock price).

6.2 Trace Driven Simulation

Trace-Driven simulation is commonly used in real system and simulation results have more credibility. This type of simulation provides more accurate workload and detailed information that helps users to learn indepth the simulation model. Usually, input values in the simulation are constant. High-level detail information increases the complexity of the simulation and workloads may change. This leads to a drawback of trace driven simulation that representativeness of the simulation needs to be suspicious.

6.3 Discrete-Event Simulation

Discrete-event simulation is widely used in WSNs, because it is capable of simulating lots of jobs running on different sensor nodes. Discrete-event simulation includes some of components. This simulation can record pending events and these can be simulated by routines. The system state depicted by global variables can represent the simulation time, which allow the scheduler to predict this time in advance. Input routines, output routines, initial routines and trace routines are included in this simulation. In addition, this simulation provides dynamic memory management, i.e., new entities can be added and drop old entities in the model. Debugger breakpoints are provided in discrete-event simulation, so it is possible by the user to check the code step by step without disrupting the program operation.

7 DESCRIPTION OF SIMULATION TOOLS

The simulation tools [25] are available in two different forms. They are: (i) General Simulation Tools and (ii) Specific WSNs Simulation Tools. The general simulation tools behaviour is described using a tabular approach as shown in Table 2. Discrete Event Simulation is a trustworthy platform for modelling and simulating a different types of systems. Some of the discrete event simulators specific for WSNs are shown in Table 3.

7.1 Matlab-True Time Toolbox

The *TrueTime* toolbox which is a freeware Matlab® library. It is mainly used for simulating networked and embedded real-time control systems. Ali et al., presents [26] a new simulation methodology of WSNs. The new simulation methodology proposed by the authors proves that ability of simulink MATLAB to be a useful and flexible move towards to revise the effect of different physical layer parameters on the recital of WSNs. They use MATLAB(7.6) Simulink to build a complete WSNs system. In [27] Alberto Cardoso et al., proposes a higher level simulation platform for WSNs based on the *TrueTime* toolbox. The authors finds that *TrueTime* toolbox was indispensable for a correct, reliable and effortless abstraction of the underlying layers of the simulator, primarily the electronics and linking issues following each wireless node and the wireless network.

Simulink block library and a collection of MEX files [27][28][29] are the major components of the simulator software. The kernel block simulates a real-time kernel executing user-defined tasks and interrupt handlers. The various network blocks allow nodes (kernel blocks) to communicate over simulated wired or wireless networks. The latest release of this simulator is TrueTime 1.5. This version characteristics a couple of standalone network interface blocks that makes it simpler to develop networked control simulations. A TrueTime simulation is programmed to a great extent the same way as a real time embedded system. Matlab code or C++ can be used to write the application. The execution/transmission times must be specified by the developer , this finds the main difference from real programming. TrueTime is treated as a very flexible co simulation tool. The step from simulation code to production code is not that large in this simulation tool.

8 PERFORMANCE EVALUATIN OF SIMULATOR

8.1 Evaluation of Simulators

Recognition and Identification of the functional requirements from the simulation tool is profoundly related to the perspective in which it is to be used. The key properties of simulator which are expected from a Wireless Sensor Network application developer from excellent Wireless Sensor Networks simulator are: (i) Reusability and Extensibility, (ii) Performance and Scalability, (iii) Operating System Portability, (iv) Semantics Scripting languages, (v) Realism level of Virtual Environment, (vii) Graphics, Debug and Trace.

Table 2 General Simulation Tools

Simulation tool	Type of simulation	Programming	Availability/GUI	Details
NS-2 (current version 2.35)	Discrete Event Simulation	Object oriented extension of	Open Source/ No (provides online documents)	1.Paradigm of reusability. 2.It includes WSN specific protocols Directed Difussion

		Tcl and C++		/SMAC 3.Poor Graphical support, via Nam.
J-Sim (java sim)	Discrete Event Simulation	Java (component based) Scripting interface (Perl,Tcl and Python.)	Open Source/ Yes	Main benefit of J-sim is its considerable list of supported protocols, WSN simulation framework with a very detailed model of WSN. Implementation of localization, routing and data diffusion algorithms of WSNs.
OPNET (Provides free of charge university program for academic Research programs)	Parallel Discrete Event Simulation (OPNET Modeler Wireless suite version 16.0)	C/C++ object-oriented	Commercial/ Yes	Hierarchical model to define each aspect of the system. OPNET has ESD (External System Domain) for communicating with external software and systems.
OMNeT++	Discrete Event Simulation	C++(Highly portable with Windows,Linux and Mac OSX	Non commercial licence, (OMNESET) commercial licence/Yes	OMNeT++ is not simulator, but it provides a frame work and tools to write simulations.
EYES	framework for OMNeT++ is written for self organizing an collaborative energy-efficient sensor networks			
Mobility Framework	Basic support for Mobile and Wireless Networks. Includes some basic layers such as MAC layers (Aloha, CSMA) and network layers (flooding) as well as some basic mobility functionality and some basic application layer.			
GloMoSim (Qualnet) (Parallel computing lab @UCLA)	Parellel Discrete Event Simulation	Parsec (Simulation language Derived from C)	Commercial/ Yes (commercial version @2000 sQualNet)	Several proposals for WSNs protocols have been tested with it. Recently, a development kit for WSNs has been released, sQualnet
Ptolemy II	different models of simulation paradigms (e.g. continuos time, dataflow, discrete-event).	Java packages (component-based design of J-Sim.)	Open source/GUI	addresses the modeling, simulation and design of concurrent, real-time, embedded systems. VisualSense is a modeling and simulation framework
JiST/SWANS	Discrete Event Simulation (Virtual Machine based Simulation(JVM))	Java Byte Code	Open Source/The only graphical aid is an ***event logger.*** **Jython** is used as a scripting engine	The main drawback of JiST tool, is the lack of enough protocol models. It provides an *ad-hoc* network simulator called **SWANS**, built atop

				JiST engine and with a reduced protocol support..
NCTUns2.0	Discrete Event Simulator	kernel of a UNIX machine(embedded)	unknown	Adding WSN simulation modules to architecture is not a straightforward task.
SSFNet (Scalable Simulation Framework)	Discrete Event Simulator	Domain modeling language (APIs in JAVA as well as C++)	Open Source/GUI	DartmouthSSF (DaSSF) is a C++ implementation of SSF oriented to (parallel) simulation of very large scale communication networks.
SWAN	A specific extensions oriented towards *ad hoc* networking . SWAN is being extended to be able to execute TinyOS code, a new framework called TOSSF .			
TrueTime toolbox (MATLAB)	Networked and embedded real-time control systems	Matlab programming	Freeware/YES (XML configuration file,3Dsimulation and visualization as well as the possibility of real time simulations embedding real motes.	supports the *ZigBee* protocol.

Table 3 Specific Simulation Tools

Name of Simulation tool	**GUI**	**Type of availability**	**Programming or Operating systems**	**Description**
Tossim (powerful Emulator)	Yes (TinyViz)	Open Source/online documents	TinyOS	Tossim cannot correctly simulate due to the issue of energy consumption in WSNs.
Power Tossim	TinyOS simulator extends power model to TOSSIM , estimates the power consumption of each node.			
Tossf&Tython	Simulation systems complies TinyOS applications to emulate the Berkely MICA mote hardware. TOSSF enhances the TOSSIM scalability, Tython compliments TOSSIM executions adds a scripting environment to augment the simulation.			
Emstar (Emulator Built in C runs in real time)	Yes	Open Source	C	Can only apply to iPAQ-class sensor nodes and MICA2 motes. .
EmSim	simulator of the microserver environment each simulated node runs an EmStar stack and is connected through a simulated radio channel model.			
EmCee	Interface to real low-power radios, capable of generating radio emulation.			
EmSim	Simulator of the microservers environment. In EmSim simulated node runs an			

	EmStar stack and is connected through a simulated radio channel model.			
ATEMU (emulator of an AVR processor)	Yes XTADB debugging tool	Open Source And online documents	C (supports user to run TinyOS on MICA2 hardware)	Emulate different sensor nodes in homo and heterogeneous networks at the cost of high processing requirements and poor scalability.
Avrora (simulator of an AVR processor) Sensor platform Mica2 and MicaZ. Current version Beta 1.7.106	Lacks GUI Tools: profiling utilities, energy analysis tool, stack checker, control flow graph tool.	Open Source and online documents	Java (can simulate different programming code projects)	Support thousands of nodes simulation and saves execution time.
Prowler/Jprowler (simulates Berkley MICA motes)	YES	Specific for mica motes	Runs under Matlab/Java intended to optimize n/w parameters. (matlab code)	Jprowler supports pluggable radio models and MAC protocols
UWSim (for Underwater Sensor Networks (UWSN)).	purely object-oriented fashion using C# capabilities	Commercial	The software was developed on Windows XP using Microsoft. Net Framework	Simulates the acoustic network. Characteristics of underwater networks such as low BW, need for high frequency and the effect of salinity and temperature with depth are considered
Shawn (Designed to support large-scale network simulation.)	Not known	open source	Java	customizable sensor network simulator.
COOJA/MSPSim (primarily a code level simulator for networks consisting of nodes running Contiki OS.)	No GUI	unknown	Java application, interaction with compiled Contiki through Java Native Interface (JNI).	Nodes with different simulated hardware and different on-board software may co-exist in the same simulation.

An analysis of various simulators that are used for Wireless Sensor Networks is compared in Table 2. Reusability and extensibility is very good and Operating System Portability provided in NS-2, Jsim, Ptoelmy, Avrora as their programming languages are object oriented. Whereas in Opnet, Pmnet, Tossim, Emstar, Atemu, Shawn have quoted normal rates and in Sense it is too low. The users can obtain a better performance and scalability with Glomosim, JSim, Atemu and Avrora. The Semantics Scripting languages are available in Ns-2 (Tcl and OTcl), Jsim (perl, Tcl,

python), Ptoelmy, Tossim, Atemu. The Virtual environment for analysis of results is provided in Opnet and Sense. Graphical support for simulations , a debugging aid, a visual modeling and composition tool and result plotter are best provided with Opnet, Glomosim, Omnet, JSim, Ptoelmy and Avrora.

8.2 Analysis of Simulation Tools

Many of the simulators are developed for a specific modelling task in which they are accurate and appropriate. In such cases the description in Table 3 need to be extended or modified. Very high performance real time simulations that include operating system and application layer code execution delays are impossible with these presented simulators without extensions or use of external software. Commercial simulator OPNET, offers better support, maintenance and is proved simulation models. It is observed that for scalability of large networks NS-2 lags behind the other reviewed simulators. Simulation code exportation, e.g., C/C++ or TinyOS routing algorithm or application, to sensor node is easy in commercial platform dependent TinyOS emulators, e.g., TOSSIM runs most application code used for sensor nodes. In general purpose, platform independent simulators like NS-2, C/C++ and source code interchange is possible when required interfaces and libraries are included on the target platform.

JavaSim, SSFNet, Glomosim and its descendant Qualnet, which are publicly available network simulators, attempted to address problems that were absent and unexplained by NS2. Out of SSFNet, Glomosim and its descendant Qualnet, JavaSim developers realized the drawback of object-oriented design and attempted to molest this problem by developing a component-oriented architecture. But they choose Java as the simulation language, sacrificing the efficiency of the simulation. The designers have paid more attention to Parallel simulation, in SSFNet and Glomosim , with the Glomosim more focused on wireless networks. With respect to NS2 in terms of design and extensibility SSFNet and Glomosim are not advanced.

Although SENSE is a commonly used simulator it was originally created without the capability to provide visual feedback from a simulation. Christopher et al., [10] in their proposed work discuss the problem and solution for connecting the SENSE Wireless Sensor Network simulator with iNSpect, a visualization tool planned to work with NS2. Paulo et al., [7] presents a GUI application, called G-JSim, provides a WSN simulation tool with direct and automatic interface with the J-Sim simulation engine. The tool is freely available for download and accepts user input parameters, stores them into an XML file and automatically launches J-Sim to perform simulations. Further, it provides an enhancement of J-Sim functionality with a Guided User Interface for Wireless Sensor Networks that dramatically increase the user-friendliness of the simulator.

Keeping the low-power consumption, small code and data size, evolvability factors as main design criteria Kim et al., [26] developed an Evolvable Operating System (EOS) for Wireless Sensor Network applications . Memory space efficient thread management, collaborative thread communication model and network stack are mainly provided by EOS. It supports power management of microcontroller, radio transceiver and network wide time synchronization function. Evolvability is the most important feature with which the operating system itself is configurable and upgradeable.MATLAB with its given fundamental features provides researchers a very flexible simulation environment with application in a different scientific areas, such as, WSN communication protocols or distributed control over wireless networks.

9 CONCLUSIONS

In this paper, we have presented different simulators for both general purpose and specific purpose of simulation. Evolutionary adaptation in use of simulators provides the advantages in reusing well-tested ideas, developer basis, focusing on the special characteristics and the functioning of sensor nodes in Wireless Sensor Networks. Programming sensor network systems always remains a challenging task and hence the primary goal of simulation platforms is to alleviate these challenges. For a Wireless Sensor Network system, two features of simulators are extremely valuable: reproducible experimentation and dynamic environment modelling.

The simulation study explores the scalability, reusability, extendibility, availability of graphic tools/ scripting languages and performance. Unlike traditional computer systems, it is not sufficient to simulate the behaviour of the sensor network. The experience of practicality is much more required in analogous to simulators. Even though there are several simulators, still there is no one particular simulation tool that solves all the problems for any type of scenarios for a given WSNs.

REFERENCES

[1] Akyildiz, I.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E. A Survey on Sensor Networks. IEEE Commun. Mag vol. 40, pp.102-114, 2002.

[2] C Y Chong and S P Kumar, "Sensor Networks:Evolution, Opportunities and Challenges," Proceedings of the IEEE 2003, vol. 91, pp.1247-1256, 2003.

[3] Gowrishanka.S , T G Basavaraju, Manjaiah D H, Subir Kumar Sarkar ," Issues in Wireless Sensor Networks", In Proceedings of the World Congress on Engineering 2008 WCE 2008, vol I, July 2008, London, U.K.

[4] Marko Korkalainen, Mikko Sallinen, Niilo Kärkkäinen and Pirkka Tukeva," Survey of Wireless Sensor Networks Simulation Tools for Demanding Applications" Fifth International Conference on Networking and Services Survey of Wireless Sensor Networks Simulation Tools for Demanding Applications, 2009.

[5] David Curren ," Evaluation-of-Discrete-EventWireless-Sensor-Network-Simulators", http://www.docstoc.com/docs/132602032.

[6] E Egea-López, J Vales-Alonso, A S Martínez-Sala, P Pavón-Mariño, J García-Haro," Simulation Tools for Wireless Sensor Networks", Summer Simulation Multiconference - SPECTS , 2005

[7] Paulo A C S Neves, Iúri D C Veiga, Joel J P C Rodrigues,"G-JSIM – A GUI tool for Wireless Sensor Networks Simulations under J-SIM". http://netgna.it.ubi.pt/files/2008-ISCE.pdf.

[8] Shah Bhatti, James Carlson, Hui Dai, Jing Deng, Jeff Rose, Anmol Sheth, Brian Shucker, Charles Gruenwald, Adam Torgerson and Richard Han "MANTIS OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms"ACMKluwer Mobile Networks and Applications (MONET) Journal, Special Issue on Wireless Sensor Networks, August 2005.

[9] Masaaki Takahashi, Basit Hussain, Bin Tang ," Design and Implementation of a Web Service for Lite Os-based Sensor Networks", In Proceedings of the 5th Annual GRASP Sympsosium, Wichita State University, 2009.

[10] Harsh Sundani, Haoyue Li, Vijay K. Devabhaktuni, Mansoor Alam and Prabir Bhattacharya" Wireless Sensor Network Simulators A Survey and Comparisons" International Journal of Computer Networks (IJCN), 2(5):249.

[11] David Curren ,"A Survey of Simulation in Sensor Networks" University of Binghamton,NY,2005.

[12] C Mallanda, A Suri, V Kunchakarra, S S Iyengar, R. Kannan and A. Durresi, "Simulating Wireless Sensor Networks with OMNeT++ ",LSU Simulator, Version 1, 01/24/2005.

[13] Fei Yu, " A Survey of Wireless Sensor Network Simulation Tools" http://www1.cse.wustl.edu/~jain/cse567-11/ftp/sensor/index.html

[14] Chiara Buratti, Andrea Conti, Davide Dardari and Roberto Verdone" An Overview on Wireless Sensor Networks Technology and Evolution", ISSN 1424-8220, Sensors 2009, vol.9, 2009

[15] Lander Casado and Philippas Tsigas, "ContikiSec: A Secure Network Layer for Wireless Sensor Networks under the Contiki Operating System".
[16] Qing Cao, Tarek Abdelzaher John Stankovic,Than He" LiteOS:Towards Unix-likeAbstractions for Wireless Sensor Networks" Proceedings of ACM/IEEE IPSN, pp 233-244, 2008.
[17] Daniele De Caneva, Pier LucaMontessoro and Davide Pierattoni "Operating Systems for Wireless Sensor Networks: An Overview "Ubiquitous Computing and Communication Journal Volume 3(4).
[18] Masaaki Takahashi, Basit Hussain, Bin Tang ," Design and implementation of a web service for Lite Os-based Sensor Networks", Proceedings of the 5th Annual GRASP Sympsosium, Wichita State University , 2009.
[19] Daniel Willmann "Contiki - A Memory-Efficient Operating System for Embedded Smart Objects", 2009.
[20] Hai-ying Zhou, Kun-mean Hou, Christophe DE Vaulx and De-Cheng Zuo" A Hybrid Embedded Real-time Operating System for Wireless Sensor Networks" Journal of Networks, 4:(6), August 2009.
[21] Vlado HandZizki,Andreas Kopke, Holger karl,Adam Wolisz "A Common WSN Architerure?" In Proc.1 Gi/ITG Fachgesparach "sensornetze"(Technical Report TKN-03-012 of the Telecommunications Networks Group, Technische Universitat Berlin) ,pp.10-17, Berlin, Jul 2003.
[22] Hsung-Pin Chang, Yu-Chieh Lin and Da-Wei Chang," An Online Reprogrammable Operating System for Wireless Sensor Networks", Journal of Information Science and Engineering, 27, 261-286 (2011)
[23] Hyoseung Kim and Hojung Cha "Towards a Resilient Operating System for Wireless Sensor Networks 'USENIX Association Annual Tech '06: 2006 USENIX Annual Technical Conference
[24] www.palisade.com/risk/monte_carlo_simulation.asp
[25] Abdelrahman Abuarqoub, Fayez Al-Fayez, Tariq Alsboui, Mohammad Hammoudeh, "Simulation Issues in Wireless Sensor Networks: A Survey," SENSORCOMM 2012 : The Sixth International Conference on Sensor Technologies and Applications.
[26] Tu-Thydo,Daeyoung G Kim,Tomas Sanchezlopez,Hyunhak Kim,Seongki Hong,Minh-Long Phan,"An evolvable Operating system for Wireless Sensor Networks".
[27] Qutaiba Ibrahem Ali, Akrm Abdulmaowjod, Hussain Mahmood Mohammed,"Simulation and Performance study of Wireless Sensor Network using MATLAB," First IEEE International Conference on Energy,Power,and control,pp 307-314,2010.
[28] Alberto cardoso, Sergio Santos, Amaneio Santos and Paulo Gill, "Simulation platform for Wireless Sensor Networks based on the True Time Toolbox".
[29] Prof. Satish K. Shah, Ms. Sonal J. Rane, Ms. Dharmistha Vishwakarma,"A Simulation Study of Behavior of Wireless Motes With Reference To Parametric Variation," International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 1, issue 2, August 2012

AUTHORS

Leelavathi G is Assistant Professor in the Department of Electronics and Communication Engineering at Govt. SKSJTI, Bangalore, India. She received her B.E and M.E degrees in Electronics and Communication Engineering from Bangalore University and Visvevaraya Technological University, respectively. Her research areas are Wireless Sensor Networks and Reconfigurable Embedded systems. She has published around 10 papers in national and international conferences.

Shaila K is Professor and Head in the Department of Electronics and Communication Engineering at Vivekananda Institute of Technology, Bangalore, India. She received her B.E and M.E degrees in Electronics and Communication Engineering from Bangalore University, Bangalore. She has written a book on Digital Circuits and Systems. She obtained her Ph. D in the area of Wireless Sensor Networks from Bangalore University.She has published 40 research papers in refereed International Journals and conferences.

Venugopal K R is currently the Principal, University Visvesvaraya College of Engineering, Bangalore University, Bangalore. He obtained his Bachelor of Engineering from University Visvesvaraya College of Engineering. He received his Masters degree in Computer Science and Automation from Indian Institute of Science Bangalore. He was awarded Ph. D. in Economics from Bangalore University and Ph. D. in Computer Science from Indian Institute of Technology, Madras. He has a distinguished academic career and has degrees in Electronics, Economics, Law, Business Finance, Public Relations, Communications, Industrial Relations, Computer Science and Journalism. He has authored 35 books on Computer Science and Economics, which include Petrodollar and the World Economy, C Aptitude, Mastering C, Microprocessor Programming, Mastering C++ etc. He has over 350 research papers to his credit. His research interests include Computer Networks, Parallel and Distributed Systems, Digital Signal Processing, Digital Circuits and Systems and Data Mining.

L M Patnaik is the honorary professor in IISc. He wasVice Chancellor, Defence Institute of Advanced Technology, Pune, India. He was a Professor from 1986 to 2008 in the Department of Computer Science and Automation, Indian Institute of Science, Bangalore. During the past 40 years of his service at the Institute he has over 800 research publications in refereed International Journals and Conference Proceedings. He is a Fellow of all the four leading Science and Engineering Academies in India; Fellow of the IEEE and the Academy of Science for the Developing World. He has received twenty national and international awards; notable among them is the IEEE Technical Achievement Award for his significant contributions to high Performance Computing and Soft Computing. His areas of research interest have been Parallel and Distributed Computing, Mobile Computing, CAD for VLSI circuits, Soft Computing and Computational Neuro-science.