TinyOS-New Trends, Comparative Views, and Supported Sensing Applications: A Review

Muhammad Amjad, Muhammad Sharif, Muhammad Khalil Afzal, and Sung Won Kim

***Abstract*— The wireless sensor network (WSN) is an interesting area for modern day research groups. Tiny sensor nodes are deployed in a diversity of environments but with limited resources. Scarce resources compel researchers to employ an operating system that requires limited memory and minimum power. Tiny operating system (TinyOS) is a widely used operating system for sensor nodes, which provides concurrency and flexibility while adhering to the constraints of scarce resources. Comparatively, TinyOS is considered to be the most robust, innovative, energy-efficient, and widely used operating system in sensor networks. This paper looks at the state-of-the-art TinyOS and the different dimensions of its design paradigm, programming model, execution model, scheduling algorithms, concurrency, memory management, hardware support platforms, and other features. The addition of different features in TinyOS makes it the operating system of choice for WSNs. Sensing nodes with TinyOS seem to show more flexibility in supporting diverse types of sensing applications.**

***Index Terms*— Wireless sensor networks, operating system, sensor nodes, energy efficiency.**

I. Introduction

SENSING nodes in wireless sensor networks (WSNs) are smaller in size, compared to other nodes in conventional networks. These nodes sense their environment, process the sensed data and then transmit that data to their destination [1]. Among sensor nodes, mutual coordination and an exchange of huge amounts of information can be witnessed. Rapid progression in micro-electro-mechanical systems (MEMS) has made it much easier to deploy sensor nodes in a wide variety of situations, such as battlefields, habitat monitoring, forecasting the weather, health, mechanics, transportation, underwater sensing, ecological sensing and other military applications [2]–[5]. The sensors, being very small in size, have a limited battery life and resources. These limitations are taken into consideration when designing communications models, network topologies, various algorithms and operating systems (OSs) for tiny sensing nodes [6].

Manuscript received October 29, 2015; accepted January 8, 2016. Date of publication January 20, 2016; date of current version March 16, 2016. This work was supported by the Basic Science Research Program within the Ministry of Education through the National Research Foundation of Korea under Grant NRF-2015R1D1A1A01058751. The associate editor coordinating the review of this paper and approving it for publication was Dr. Amitava Chatterjee. *(Corresponding author: Sung Won Kim.)*

M. Amjad, M. Sharif, and M. K. Afzal are with the Department of Computer Science, COMSATS Institute of Information Technology, Islamabad 47040, Pakistan (e-mail: amjadbhutta0706@gmail.com; muhammadsharifmalik@yahoo.com; khalil_78_pk@yahoo.com).

S. W. Kim is with the Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, Korea (e-mail: swon@yu.ac.kr).

Digital Object Identifier 10.1109/JSEN.2016.2519924

WSN researchers have help with different coding parameters and various languages in providing an OS for the proper functioning of sensing nodes [7], [8]. Different operating systems are now in place for sensor nodes, but Tiny Operating System (TinyOS) is acknowledged as the most suitable one to operate in a resource-starved network like a WSN. Tiny sensing motes operate in a variety of fields. These nodes are equipped with even more limited power resources. Replacement of batteries incurs serious overhead. Therefore, one requirement is that an energy-efficient OS must be designed for these sensing nodes. TinyOS is especially designed for low power sensing motes. It was first developed as a research project, but is now acknowledged as an open OS for sensing motes [9]. There are four main requirements that compelled researchers to come out with novel, flexible and concurrent versions of TinyOS for sensing motes.

1) *Limited Resources:* Sensing motes with limited resources and smaller sizes have very limited physical and logical resources to carry out their sensing operations. A processor of usually 1-MIPS (million instructions per second) with very small memory is used in these tiny sensing motes. New advances in sensing technology are made by taking into consideration these requirements.
2) *Reactive Concurrency:* Sensing motes sense the data and then process that data. During processing, some type of data aggregation or compression is performed. After processing, data are transferred to other nodes or a base station (BS). From the BS, the data are utilized for further analysis. For all these operations, it is a requirement for the OS of the sensing nodes to be highly concurrent. Reactive concurrency enables the OS to handle real-time tasks for the sensing operation.
3) *Low Power:* Sensing motes are installed in various locations. Their replacement is not an easy task. Therefore, changing or charging the batteries incurs serious overhead. For sensing motes to operate untethered, a continuous power supply is mandatory. Therefore, TinyOS was designed by taking into consideration the limited power of sensing motes. TinyOS is not only an energy-efficient OS; it also helps other sensing applications to conserve energy in their sensing operations.
4) *Flexibility:* It is necessary for the OS of sensing motes to be flexible enough to support novel and diverse sensing applications. TinyOS supports modularity and a large number of hardware platforms.

TinyOS was first developed in 2000 in University of California, Berkeley. It was started as the research project and was used only by the researchers. In late 2000, a systematic architectural directions for TinyOS was proposed [186]. TinyOS version 0.6 was introduced in 2001, and addressed certain limitations in its programming model. In January 2002, a bootcamp was arranged for TinyOS, while the core work group for TinyOS was formed in 2004. Version 2.0 and 2.0.2 were introduced in 2006 and 2007, respectively. Safe threads were added in TinyOS in 2008. Version 2.1, and 2.1.2 became available in 2010 and 2012. Currently, TinyOS development is transformed to GitHub, where the researchers can contribute to its development. Now, there are about 35,000 downloads of this free available operating system per year [12].

A monolithic architecture, novel t-kernel integration, efficient power management, concurrency handling and supportive components for diverse types of communications make TinyOS a stable and self-contained OS. Robust systems were once categorized as difficult to write, but evolving language extensions in TinyOS have made it an OS for embedded systems. TinyOS has made its way into well-known computing projects, such as Cisco's smart grid systems and Xen [11]–[13]. This paper encompasses the detailed features of TinyOS, its architectural and component models, their development, and the main advantages it added in making sensing node operations more and more reliable, flexible and robust. Our study has taken into consideration all possible aspects of TinyOS, from the programming model to its supported sensing applications. It is the first study of TinyOS that shows new trends and its novel supportive sensing applications. Table I defines the abbreviations used extensively in the paper, while Table II shows a comparison of this study and already existing surveys on TinyOS and other OSs for WSNs.

The rest of the paper is organized as follows. Section II describes the programming model of TinyOS, which consists of concurrency and execution. Section III surveys the scheduling algorithms used for TinyOS. Section IV and Section V cover the memory and energy management techniques of TinyOS. In Section VI, energy management by TinyOS, especially in the communications process with reference to communications protocols, is discussed. Section VII describes the simulators of TinyOS. A detailed comparative view of TinyOS with other OSs for WSNs is given in Section VIII. Section IX looks at the TinyOS-supported hardware platforms. Section X lists the TinyOS-supported sensing applications, while in Section XI, limitations and modifications of TinyOS are broadly discussed.

TABLE I
ABBREVIATIONS

Symbol	Description
ADRS	Adaptive Double Ring Scheduling
ASVM	Application Specific Virtual Machines
BS	Base Station
CHs	Cluster Heads
CSP	Communicating Sequential Processes
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CTP	Collection Tree Protocol
DS	Deadline Scheduler
DSVR	Destination Sequence Vector Routing
DHCP	Dynamic Host Configuration Protocol
EDF	Earliest Deadline First
EATT	Energy-Aware Target Tracking
FIFO	First In First Out
HAL	Hardware Adaptation Layer
HIL	Hardware Interface Layer
HPL	Hardware Presentation Layer
ICEM	Integrating Concurrency Control and Energy Management
IoT	Internet of Things
IPFIX	IP Flow Information Export
LAD	Location-Aided Routing
LEACH	Low-Energy Adaptive Clustering Hierarchy
MAC	Medium Access Control
MEMS	Micro-Electro-Mechanical Systems
OS	Operating System
QoS	Quality of Service
RTS	Real-Time Scheduling
SANETs	Sensor Actuator Networks
SOSANETs	Service-Oriented Sensor Actuator Networks
SPIN	Sensor Protocol for Information via Negotiation
TCP	Transmission Control Protocol
TinyOS	Tiny Operating System
TinyOS LPL	Tiny Operating System Low-Power Listening
TOSThread	Tiny Operating System Threads
TOS-PRO	Tiny Operating System Preemptive Original
TOSSTI	Tiny Operating System with Software Thread Integration
TORP	Tiny Operating System Opportunistic Routing Protocol
TOSSF	Tiny Operating System Scalable Simulation Framework
TOSSIM	Tiny Operating System Based-Simulator
UTOS	Untrusted Extensions for Tiny Operating System
WSNs	Wireless Sensor Networks

II. Programming Model

Programming TinyOS for tiny sensing nodes has various constraints. The scattered and distributed nature of nodes confronts the programmer. Different programming models have been adopted to provide an OS for sensing nodes. Choosing a particular programming model mainly depends upon three attributes of WSNs: first, the nature of the sensing nodes; second, what tasks the nodes are going to perform within the group; and third, the network type [14].

An OS for sensing nodes should be more collaborative, fault tolerant and futuristic. The programming model for TinyOS follows in the footsteps of component-based programming models. One of the dialectics of C, commonly known as NesC, is behind the programming of the novel TinyOS for WSNs. The main module of NesC consists of an editor, a parser, a model generator, a simulator and a model checker. These modules of the NesC architecure are shown in Fig. 1 [15].

The whole programming model of TinyOS is a combination of different components [16]. These components, when categorized, fall into three abstractions: commands, events and tasks [17]. A command initiates a component to perform some type of operation, which is then narrated into a request message. An event component displays output.

TABLE II
COMPARISON BETWEEN THIS STUDY AND AVAILABLE SURVEYS

Approaches	Key Concepts	This Study	Phani et al [11]	Farooq et al [10]	Strazdins et al [20]	Dong et al [8]
Programming Model	Interfaces	✓	✓	✓		✓
	Components	✓	✓		✓	✓
	Concurrency	✓	✓	✓		✓
Scheduling	FIFO	✓	✓	✓	✓	✓
	Other Advanced Scheduling Approaches	✓				
Energy Conservation Approaches	TOSSTI	✓				✓
	HPLP	✓	✓			
	Energy Tracking	✓				
TinyOS-based Energy Efficient Network Communications	Specialized TinyOS Transport Layer Protocols	✓				
	TinyOS-supported Routing Protocols	✓		✓		
	TinyOS-supported MAC Protocols	✓		✓		
	TinyDB	✓			✓	
	TinyOS Support of IP and IPv6	✓			✓	
	TinyOS for ZigBee	✓	✓			
	TinyLTS	✓				
	TinyOS for Heterogeneous Networks	✓	✓			
Simulators	TOSSIM (Basic TinyOS Simulator)	✓	✓			
	Other Advanced Simulators	✓	✓			
Comparative View of TinyOS with Other WSNs OSs	Other OSs Include Contiki, LiteOS, SOS, MANTIS, Nano RK	✓	✓	✓		
TinyOS-based Supported Sensing Applications	Basic Sensing Applications	✓			✓	
	Advanced Applications	✓			✓	
Advanced Versions of TinyOS	TinyOS 2.0	✓		✓		
	TinyWifi	✓				
	Dynamic TinyOS	✓				

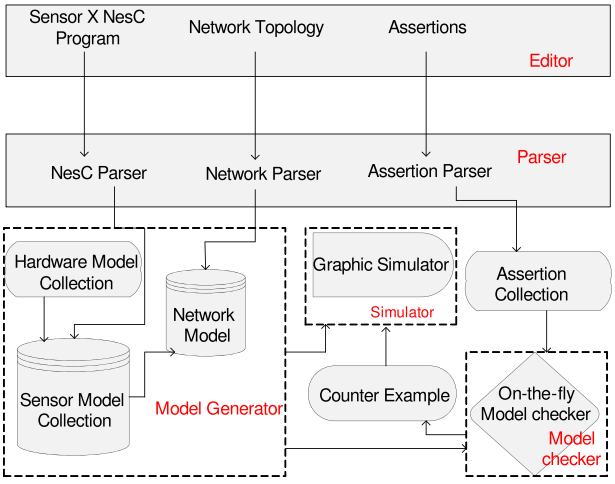

Fig. 1. Architecture of NesC (adapted from [15]).

Communication between the components is achieved with the help of tasks. Components are also provided with the interfaces. These component interfaces fall into two classes.

1) Interfaces that the component uses.
2) Interfaces the component provides.

Interfaces also use a bidirectional feature for their operation. In addition to these component-based interfaces, TinyOS has other interfaces for its various operations [18].

Contracts [19] are now employed for the interfaces, which are used again and again. Hence, the components that are reused by the applications are now replaced by the interface contracts [21]–[23]. Many hardware abstractions were also added to TinyOS [24], [25]. Using NesC and then joining the components is not an easy task in coding for TinyOS. Linking models are apparently not the same for C and its dialectics [26], [27]. Different design patterns have been introduced for software to evolve a programming model for TinyOS in NesC [28]. The programming model of TinyOS in NesC mainly focuses on concurrency and execution of tasks. These two features and their support in TinyOS are discussed below.

A. Concurrency Support

In NesC programs, components of user applications not only interact with TinyOS code but also with one another.

This has been achieved with inclusion of concurrency in TinyOS. TinyOS follows an event-driven concurrency model [29], [30].

The event-driven concurrency model in TinyOS sometimes introduces certain complexities in the normal operation of sensing nodes. As sensing nodes have to operate undeterred, blocking certain components of an application can block the whole sensing operation. Race condition in the NesC compiler has already been integrated to mitigate concurrency-related problems. Application programmers now have multiple techniques for checking concurrency-related errors, out of which a process algebra known as communicating sequential processes (CSP) [31] is highly used. Another method to achieve maximum concurrency with scarce processing resources is an implementation of TinyOS threads (TOSThreads). The TinyOS concurrent model follows a synchronous and asynchronous model of execution. In TOSThreads, threads are categorized into application-level threads and kernel-level threads. Kernel-level threads are assigned with high priority and cannot be preempted by user-level threads. User-level threads cannot interact with OS-level threads, either synchronously or asynchronously [32], [33].

TOSThreads are used to change the conventional TinyOS non-preemptive behavior to preemptive behavior. The addition of these TOSThreads brings extra complexity to TinyOS. Due to this complexity, the component-based programming model with preemptive scheduling lost its efficiency. To address this issue in TOSThreads, another preemptive technique has been added; that approach is the TinyOS preemptive original (TOS-PRO) approach. The main benefit of TOS-PRO is enhanced concurrrency in TinyOS. This improvement in concurrency results in system improvements, which finally results in fast execution of real-time tasks [34].

Enhancing concurrency has been an interesting area in WSN OS design. To improve concurrency, lightweight, thread-like abstractions, called fibers in concurrency modeling, are now also seen in TinyOS [35]. In recent developments, the concurrency information of expected applications is obtained before execution via Integrated Concurrency and Energy Management (ICEM) in device drivers [36].

B. Execution Model

TinyOS written in NesC has an execution model that is sensitive to interrupts. Different computational tasks are executed non-preemptively and follow the pattern of run to completion. However, the main hurdle in execution of such tasks is the occurrence of interrupts in the execution of tasks [15]. TinyOS supports different types of hardware platform in the WSN domain. These different platforms introduce their own interrupts relating to their hardware designs. Software interrupts have little overhead in execution of tasks [37].

Tiny sensing nodes with limited processing capability have to time-share a processor among applications, the OS and different communications protocols. Better execution and fast convergence in this case can be achieved when individual components are virtually partitioned. An event-driven execution model produces maximum concurrency that works within the limited resources, like energy and memory. Introduction of threads to gain more execution and concurrency also demands more memory [38], [39]. Now researchers are finding ways to predict memory and power demands by applications before they make their way into real sensing networks [40]. Different virtual machines are now in place to predict the future demands of user applications and communications protocols for memory and execution time. Application-specific virtual machines (ASVM) can go to the extent of reprogramming already deployed WSNs [41], [42]. Different visualizing toolkits for TinyOS have also been constructed to assist programmers. With these toolkits, execution time of different tasks can be predicted. In [43], a toolkit for TinyOS 2.0 was developed to predict the runnng time of different embedded applications.

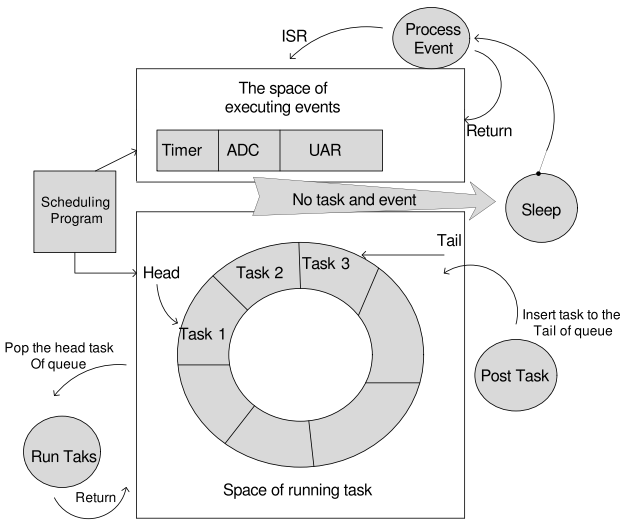

Fig. 2. TinyOS scheduler (adapted from [44]).

III. Scheduling Algorithms

TinyOS programmed in NesC is equipped with the basic components of events and tasks. In the initial versions of TinyOS, there was a single type of task. Therefore, a wide variety of scheduling algorithms was not needed. A simple scheduler was integrated to assist the single task dependent TinyOS. A basic task scheduler for TinyOS was run to complete the task scheduler. It was a non-preemptive task scheduler. The basic TinyOS scheduler is shown in Fig. 2 [44]. This earlier form of task scheduler was first in, first out (FIFO) [8], [10], [11], [17], [20], [44], [45]. Now, WSNs have found their applications in multiple fields. Therefore, a wide variety of tasks have to be handled by the OS. With the increasing number of tasks, the number of scheduling algorithms has also increased. FIFO is now not the only scheduling parameter in TinyOS. Multiple scheduling techniques have now been integrated [44].

A. Priority Scheduling

In this technique, tasks are given priority, and that priority is based on their importance. Real-time and other network packets are now given the highest priority to ensure quality of service (QoS). TinyOS with priority scheduling is discussed elsewhere [46]–[48].

B. Earliest Deadline First (EDF)

In this scheduling technique, real-time sensitive traffic is scheduled. Tasks are now prioritized, depending on their remaining execution times. TinyOS with EDF [44], [47], [48] shows enormous responsiveness towards time-critical data.

C. Real-Time Scheduling (RTS)

Another scheduling algorithm to handle real-time tasks and network packets is the real-time scheduling mechanism. This method employs a pre-emptive technique to execute a given task in TinyOS [49], [50]. Real-time tasks can also be scheduled by classifying them into periodic and aperiodic tasks. These periodic tasks, known as time-bounded tasks are then executed by the periodic scheduler, and aperiodic tasks can be scheduled with a time-unbound scheduler known as the aperiodic scheduler. Response time for aperiodic tasks has also improved many times over. In this category of real-time scheduling, energy is conserved many times, as compared to other scheduling techniques used in TinyOS [48], [51].

D. Deadline Scheduler (DS)

This is an enhanced version of FIFO scheduling. However, deadline is a new parameter added to the TinyOS scheduling technique. Incoming tasks are now categorized based on their deadlines [47], [48].

E. Priority-Based Soft Real-Time Scheduling

Certain tasks take a lot of time in their execution. This can lead to overloading. In this situation, real-time tasks are not executed properly. To mitigate this problem, priority-based soft real-time scheduling was introduced in TinyOS for smooth execution of real-time tasks [47], [48], [50].

F. Job: A New TinyOS Based Task Scheduler

An earlier version of the TinyOS task scheduler was run to completion, non-preemptive. This introduced a problem for larger tasks because these tasks had to wait for a long time, which reduces system responsiveness. This issue was addressed with cooperative and multithreading multitasking. Job is a task scheduler for TinyOS, which incorporates cooperative and multithreading multitasking approaches for executing the larger tasks in systems [52].

G. Adaptive Double-Ring Scheduling (ADRS)

In this TinyOS scheduling method, there are two types of task cycle queue. One task cycle queue is given higher priority than the other. Real-time tasks can also be executed in ADRS, because they are placed in a task cycle queue that has a higher priority. ADRS is simulated in the TinyOS-based simulator (TOSSIM). The simulation results showed that TinyOS using ADRS provides better performance [48].

H. Co-Routine Scheduling

Multitasking is incorporated into TinyOS with the help of a co-routine scheduling mechanism. Tasks are labelled as routine, and each routine has its own stack. This method of having a separate execution stack is more similar to having threads in execution [48].

With the growing number of supported sensing applications in TinyOS, scheduling techniques have also been modified. Different routing techniques also introduced new packet scheduling methods to facilitate better convergence and QoS [53]. Scheduling the power in the network is also endorsed as the main scheduling research domain [54]. Details of the above-mentioned scheduling policies with their advantages and limitations are given in Table III.

IV. Memory Management and Protection

WSNs consisting of tiny nodes have to operate with limited resources. The available processing power and memory are not enough. Hardware protection of memory is not available in tiny nodes to safely manage the sensed and processed data. Earlier versions of TinyOS only supported static memory allocation due to limited available space [55]. However, gradual revisions, and new enhancements in TinyOS, provide enhanced features like memory safety and memory safety checks. TinyOS 2.0 introduced more memory safety, compared to simple TinyOS 1.0 [56]. TinyOS 2.0 is used as a basis for further enhancements to provide more protection of memory. "Safe TinyOS" was developed with a main function of providing memory safety to tiny nodes. Various memory checks formed a red line for safe execution of tasks. This red line prohibits bogus and unsafe programs from executing and, in this way, "Safe TinyOS" provides maximum memory safety for sensing nodes [57].

Untrusted extensions for TinyOS (UTOS) is another revision in TinyOS editions for providing more memory safety, compared to "Safe TinyOS." In UTOS, untrusted execution of data is isolated first and then terminated. Simpler modifications are required to transform TinyOS into UTOS. Migration from simple TinyOS to UTOS gives more memory safety, as depicted in Fig. 3 [58]. The size of the OS for WSNs can also be minimized with the help of different programming paradigms. The introduction of protothreads enabled OS developers to write the code for WSN OSs with the fewest possible lines [59]. This reduction in code length demands less memory, which is the core demand for programming a sensing-node OS.

There was also the introduction of UnStacked C in TinyOS. In this approach, applications of a sensornet that support TOSThreads can be modified in such a fashion that they can easily be transformed into stackless threads during the building process. The UnStacked C approach makes TinyOS memory-efficient and conserves energy [60]. Dynamic TinyOS provides the user with a dynamic auto-update feature in TinyOS and its components, without interrupting the operations of sensing nodes [61].

V. Energy Management in TinyOS

Battery-operated tiny sensing nodes are widely distributed to sense the required data. Replacing the batteries incurs extra overhead on a resource-constrained network. Therefore, every feature of WSNs is taken into consideration while

TABLE III
TinyOS SCHEDULING POLICIES: AN OVERVIEW

Relevant References	Scheduler	Type	Concept	Advantages	Limitations
[8], [10], [11], [17], [20], [44], [45]	FIFO	Non-Preemptive	First come, first served	Simple tasks can easily be scheduled	Longer tasks have to wait, which causes degradation of system responsiveness
[46], [47], [48]	Priority Scheduling	Preemptive/Non-Preemptive	Critical tasks are given high priority	It can ensure QoS	Incurs overhead in resource-scarce sensing nodes processors
[44], [47], [48]	EDF	Non-Preemptive	Priority based on execution deadline of tasks	Dynamic decisions of priority	Real -time scheduling is not supported
[48], [49], [50], [51]	RTS	Preemptive	Involves many other scheduling policies	Real-time traffic is handled	Other complex tasks introduce complexities to the system
[47], [48]	DS	Non-Preemptive	Remaining execution deadline is used for priority of tasks	Increases throughput and resolves overload	Low-priority jobs suffer more
[47], [48], [50]	Priority-Based Soft Real-Time Scheduling	Preemptive/Non-Preemptive	Real-time Tasks are assigned high priority	Increased execution of real-time tasks	Tasks other than real-time tasks suffer more
[52]	Job: Scheduling Policy	Preemptive	It is a multithreading and multitasking approach	It was designed for execution of long tasks	Collisions and message loss can be seen
[48]	ADRS	Preemptive/Non-Preemptive	Two queues are maintained with different priorities	Two queues help execute longer, smaller, and real-time tasks more efficiently	More memory needed which burden the resource scarce sensing system
[48]	Co-Routine Scheduling	Preemptive	Multitasking is incorporated in this scheduling approach	Threads are easily managed	Separate execution stack creates serious overhead

designing them [62]. Different power-saving techniques have been introduced to conserve maximum power. Sensing nodes in some cases are shut down when they are no longer needed for sensing data. Power is dynamically distributed in the whole sensing network [63]. Usually, energy conservation is focused in three major operations of the sensing nodes: processing, communications, and sensing [64], [65].

Different techniques have been incorporated in TinyOS to achieve minimum power utilization. TinyOS with software thread integration (TOSSTI) is a method in which energy is conserved in TinyOS. By the integration of software threads, TinyOS makes efficient use of idle time during transmission, processing and sensing of data [66]. In TinyOS that supports high-power listening (HPL), TinyOS estimates the overall load of the sensing nodes and then dynamically allocates the required energy to the sensing nodes [67]. This can only be possible with accurate estimation of energy consumption in the sensing nodes. Sensing nodes consume energy in a variety of ways [68], [69]. TinyOS, in this case, is the most efficient OS because it estimates the energy consumed by the sensing nodes, by TinyOS itself, and by its components. TinyOS supports various methods of estimating the energy consumption for different applications. One of the methods is the energy tracking system.

In this method, energy-tracking components are added in the TinyOS programming model. These energy-tracking components track the energy consumption of various components in the sensing nodes. Energy tracking of the processor, the transmission module and the sensing module is done with the help of TinyOS. The energy-tracking method is shown in Fig. 4 [70], [71]. Different energy conservation techniques have been implemented using TinyOS. One of the techniques, called energy-aware target tracking (EATT), is used to track the energy consumption and was developed only for TinyOS [72].

VI. TinyOS AND ENERGY-EFFICIENT COMMUNICATIONS

Sensor nodes in WSNs sense data and then transfer them to the BS. This communication is designed to be energy-efficient to save the maximum amount of energy in the system. Signal propagation, reception, packet transmission, idle and sleep behavior of sensing motes are modelled to be energy-efficient [73]. The OS in this regard plays a crucial role. TinyOS, with its advanced components, favors energy-efficient communications. Different mechanisms are supported by TinyOS to estimate the consumption of energy. Accurate consumption of energy for communications helps to estimate the network lifetime and the stability of the whole network. TinyOSs supportive behavior for energy-efficient WSN communications (with relevant references) is shown in Fig. 5.

A. TinyOS Support for Communications Protocols

TinyOS provides compatibility for energy-efficient protocols. A large number of communications protocols are well-supported in TinyOS. Protocols at the medium access control (MAC) layer, transport layer and network layer are specifically designed to consume less power. TinyOS, being the most widely used OS for WSNs, is so flexible that it supports the maximum number of energy-efficient protocols.

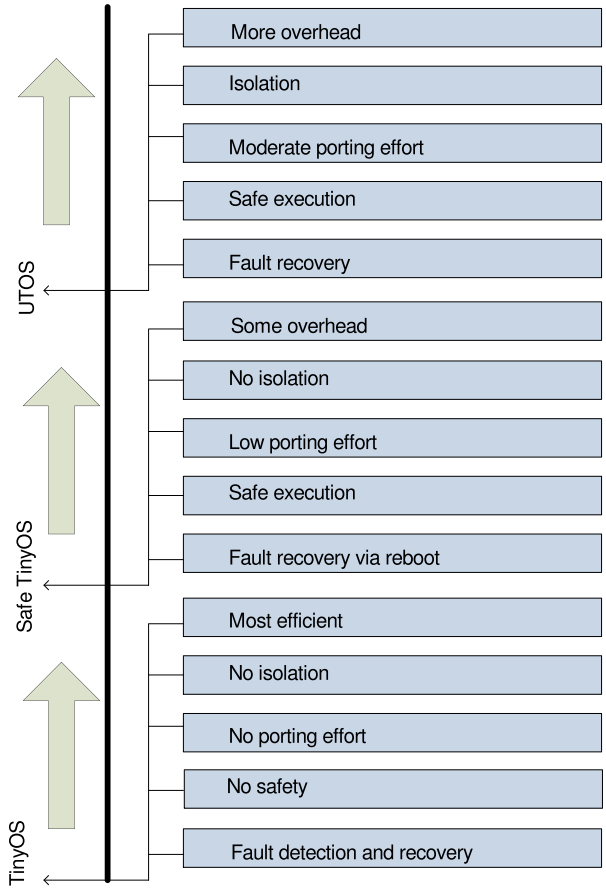

Fig. 3. Transition of TinyOS from safe to untrusted extensions (adapted from [58]).

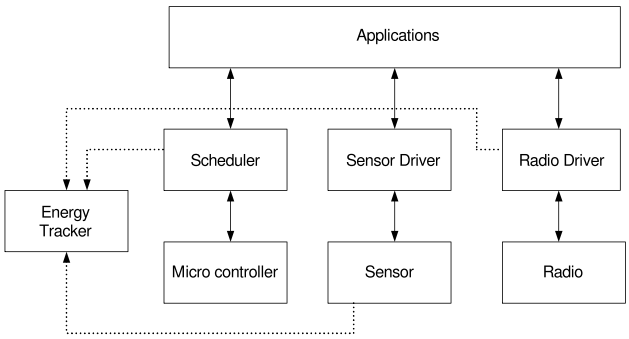

Fig. 4. Energy-tracking technique by TinyOS (adapted from [71]).

1) TinyOS Support for Transport Layer Protocols: Protocols for the transport layer have been designed to conserve energy under the limited resources of WSNs. Protocols of the transport layer are well-supported by TinyOS. TinyOS seems to enhance the performance of protocols in the sensing operation. A hybrid, dynamic, reliable protocol was designed for the transport layer, and its performance was measured with TinyOS [74]. Another widely used transport layer protocol is the post-order based protocol. This protocol is suitable not only for the transport layer but also for the routing layer. The post-order based protocol is implemented in TinyOS, and shows the best results with this event-driven OS [75].

2) TinyOS Support for Network Layer Protocols: Instead of conventional routing protocols, energy-efficient routing protocols have been designed for WSNs. So WSN researchers have developed robust and energy-saving routing protocols [76]–[78]. TinyOS provides support for these routing protocols to save the maximum amount of energy [79].

Opportunistic routing has been introduced in WSNs. This routing approach seems to conserve more energy in WSNs [80], [81]. A very specific TinyOS-based opportunistic routing protocol, named the TinyOS opportunistic routing protocol (TORP), was proposed to conserve energy. This approach selects the forwarding nodes in a more efficient way and then forwards data to nearby nodes, and hence, conserves energy in the system. TORP has enhanced network lifetime, scalability, throughput and energy efficiency, compared to other conventional routing protocols [82]. Low-energy adaptive clustering hierarchy (LEACH) [83] is the network layer protocol for WSNs. It is a conventional routing protocol for WSNs. The LEACH protocol has been tested on TinyOS. The implementation of LEACH under TinyOS shows that LEACH performs better in conjunction with TinyOS [84]. The LEACH protocol is extensively implemented on TinyOS. It also shows better performance with the TOSSIM simulator. Other routing protocols have also been derived from the LEACH protocol and have been implemented in TinyOS [85]. Routing protocols for ad hoc networks show smooth operation with TinyOS. Comparisons of energy efficiency for different routing protocols are made by implementing them under TinyOS. Location-aided routing (LAD) and destination sequence vector routing (DSVR) were implemented using TinyOS. TinyOS supports the energy-efficient communications of these routing protocols [86]. Sensing nodes in WSNs share network traffic load with other nodes in the network. Various load-balancing routing approaches have been proposed for WSNs. A load balancing routing scheme presented by Daabaj [87] enables the sensor nodes to balance the load and provides an energy-efficient routing scheme. In this approach, the load-balancing routing algorithm forms a tree-like forwarding table and tracks the packets. Power consumption in energy-balanced routing protocols is measured in TinyOS with dynamic power scaling. With its implementation in TinyOS, dynamic power scaling provides optimal power usage during routing operations [88]. The sensor protocol for information via negotiation (SPIN) is a data-centric routing protocol for WSNs. SPIN follows the energy-efficient event-driven delivery model. This routing protocol is implemented using TinyOS and shows the best results in improved network life and stability [89].

3) TinyOS Support for MAC Layer Protocols: Different energy-efficient MAC layer protocols have been designed and tested with TinyOS. Carrier sense multiple access/collision avoidance (CSMA/CA) is extensively used in WSNs. This protocol gives optimal performance with TinyOS. Improved versions of CSMA/CA were also tested on TinyOS-supported simulators, such as TOSSIM and PowerTOSSIM [90]. B-MAC is a well-known second layer protocol that is specifically designed for TinyOS. This MAC layer protocol has a sleep procedure to stabilize the network. Energy conservation in the MAC layer enhances the overall energy of the sensing

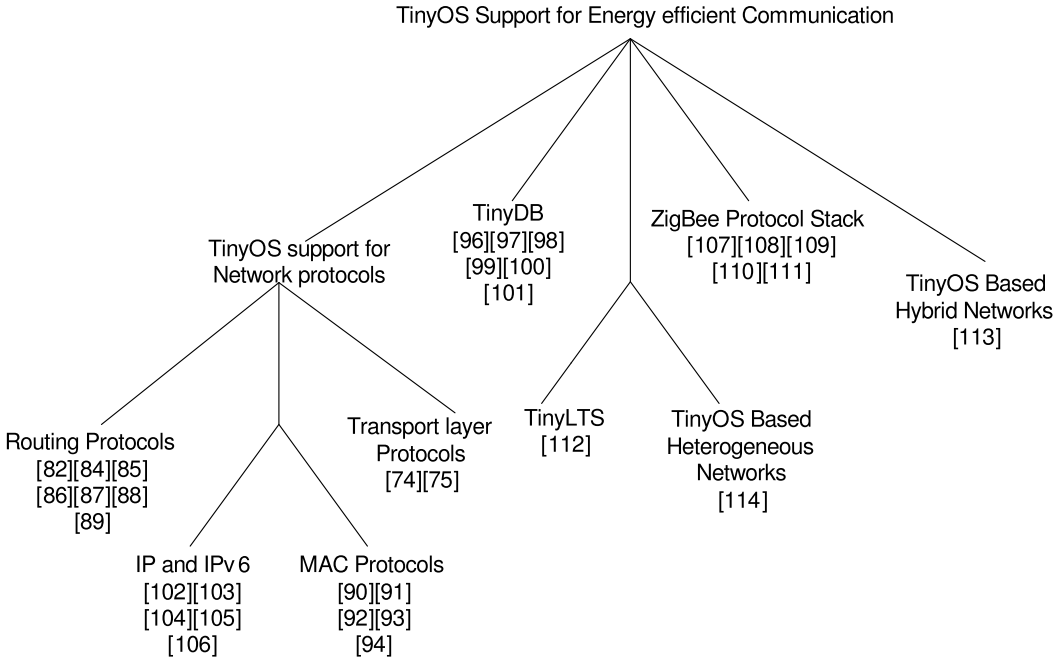

Fig. 5. TinyOS-based energy efficient communications.

nodes [91]. Another protocol in the MAC layer in TinyOS is TinyOS low-power listening (TinyOS LPL). This is an efficient energy-saving MAC protocol designed only for TinyOS [92]. X-MAC works well with TinyOS for duty cycled WSNs. In X-MAC, energy is conserved more by employing a shorter preamble. This shorter length preamble makes it the protocol of choice. It was tested with TinyOS, and TinyOS seems to be more compatible with this new version of a MAC protocol [93]. Different versions of MAC protocols introduced compatibility issues. Each MAC layer protocol is suitable for some specific application or set of applications. These issues introduce the problem of compatibility between protocols. This, in turn, badly affects communications between sensing nodes. To overcome compatibility issues, a MultiMAC protocol stack was introduced in WSNs. This MultiMAC stack is fully supported and developed under TinyOS. With the help of this approach, interoperability problems at this layer are resolved. The MultiMAC protocol stack initially comprised three well-known MAC protocols: CSMA/CA, LPL-MAC and TDMA MAC. This stack is more flexible and scalable, because it can support other MAC layer protocols, as well [94].

4) Support for IP and IPv6: Sensing motes, as compared to other network nodes, did not use the conventional internet protocol (IP) addressing scheme. However, new research has enabled tiny sensing motes to use both IP and IPv6 [102], [103]. With the help of IP communications, low-power sensing nodes using TinyOS can communicate directly with conventional IP networks. Both the IETF 6LoWPAN and RoLL research groups have come out with a new benchmark, which focuses on implementation of IPv6 in WSNs with the help of TinyOS [104]–[106].

There are certain limitations regarding the use of IP in WSNs. These limitations hindered the use of IP for WSNs. Now various modifications have been introduced in WSNs OSs architecture to make it suitable for IP. The main challenges faced by WSNs OSs for using IP are discussed as follows [228].

- Large IP header overhead is considered not suitable for tiny low power sensing nodes OS. Sensing nodes radio communication module consumes much energy while transmitting and receiving the IP packet. IP header packet size is 20 bytes for IP and 40 bytes for IPv6. To address this challenge, various header compression approaches are used by the WSNs to use IP and IPv6. TinyOS supports various header compression techniques to make the implementation of IP on WSNs [102].
- Addressing scheme of conventional IP network relies on global IP address that uses dynamic host configuration protocol (DHCP) in case of IPv4, and stateless address auto-configuration (SAA) in IPv6, which in turn, creates large overhead for low bandwidth and energy scarce sensing nodes. TinyOS seems to provide the compatibility between the data-centric routing in WSNs and address centric routing of IP networks [103].
- Compared to IP network, WSNs have very limited battery life. The replacement of battery is not an option in many cases such as battlefield implementation. IP consumes larger bandwidth for convergence and controlling the network topology compared to WSNs. Hence, various novel energy efficient routing protocols have been designed separately for WSNs with less code size. TinyOS supports an efficient energy tracking mechanism for implementation of IP in WSNs.

While residing within these limitations, various approaches have been adopted to introduce interoperability in the sensing nodes architecture and transmission control protocol and the internet protocol (TCP/IP) stack. Two approaches which are adopted for compatibility are the proxy-based and the sensor

node stack-based [95]. With the introduction of internet of things (IoT), the sensing nodes that are now termed as the data producing nodes, use the sensor stack-based approach for communication with the internet. TinyOS supports the sensor stack-based approach by employing the efficient inlined functions in its programming model. The optimization of programming model of TinyOS gives the energy efficient solution for this stack-based approach.

Interoperability of TCP/IP for sensing nodes was firsts introduced in [229] by introducing the micro IP (uIP) and lightweight IP (lwIP) TCP/IP stacks for small sensing motes. These uIP and lwIP TCP/IP stacks are tested with TinyOS. TinyOS provides energy efficient implementation of these TCP/IP stacks on sensing nodes while using the MICAz hardware platforms.

A WSNs application known as intrusion detection first used the IP-based sensing nodes for its sensing operation [230]. In this application, embedded sensor board (ESB) was used on Contiki operating system and later on TinyOS. Both the contiki OS and TinyOS give the optimal compatibility for both the address centric routing of TCP/IP protocol stack and data centric routing protocols of WSNs.

In [231], the IPv6 and IPv4 have been compared for their implementation in WSNs. Despite the 128 bit address space of IPv6, IPv6 is more flexible and has advantages when it is used for WSNs. SAA approach of IPv6 and larger address space to cover large networks make it best fit for emerging WSNs for IoT. TinyOS coding is edited with respect to SAA approach of IPv6.

To make the IPv6 compatible for TinyOS-supported sensing nodes, the 6LoWPAN working group (6LoWPAN WG) has developed an intermediate layer to incorporate IPv6 into 802.15.4 [232]. The overhead created by IPv6 header is minimized with the introduction of compression mechanis in 6LoWPAN. The header compression mechanism 1 and 01 (HC1 and HC01) are supported by TinyOS programming model. In addition to compression, the fragmentation technique also divides the IPv6 packets into several 802.15.4 frames, hence makes the IPv6 suitable for low power tiny sensing nodes.

In WSNs, various protocols communicate with one another with the help of message passing. TinyOS supports active message passing in message-oriented communications. These active messages obtain the help of "Split Control" to manage energy in the whole network [95]. TinyOS makes WSN communications reliable and fault-tolerant by supporting message-based communications. With the passage of time, TinyOS has witnessed many developments in its structure to make WSN communications energy-efficient.

B. TinyDB

In the communications process, sensing nodes using TinyOS use features of TinyDB. With the help of TinyDB, sensing motes extract useful information from the network. It is actually the query-processing system that conserves the energy in the system and made the programming task much easier. To extract the information, low-level code does not have to be written separately for TinyOS, because TinyDB with a structured query language (SQL) type of interface provides query processing. The following features were added with the integration of TinyDB in TinyOS [96]–[99].

- *Network Layout:* TinyDB manages the whole network topology. The whole communications network layout is maintained in TinyDB.
- *Efficient Query Handling:* Time-critical and non-critical data, as mentioned by Mayer et al. [100], can also be provided to end users with TinyDB.
- *Data Protection:* Sensing motes sense data and then transfer them to the BS [101]. In this transfer, data integrity and protection under TinyOS is provided with the help of TinyDB.
- *Smooth Communications:* Sensor network communications is more reliable and smoother with the help of TinyDB.

C. Support of IEEE 802.15.4/ZigBee Protocol Stack

The IEEE 802.15.4/Zigbee protocol stack is now used by sensing applications. WSNs with ZigBee applications are gaining more and more importance. TinyOS is utilized in various sensing applications with the integration of ZigBee. MICAz platforms are considered more useful when using ZigBee with TinyOS [107]–[109]. ZigBee was also tested on different hardware platforms running TinyOS [110]. Porting TinyOS from one platform to another platform with ZigBee does not introduce many complexities, compared to other protocol stacks [111].

D. TinyLTS

TinyLTS is an extension of TinyOS. Through the help of TinyLTS, network-related logging and tracing can be achieved. Through logging and tracing, network applications can be monitored and analyzed without any other separate logging tools. TinyLTS can get into the applications main components and gives an idea of their behavior. TinyLTS can also separate dynamic and static information at compile time [112].

E. TinyOS Implementation for Hybrid Networks

WSNs can be implemented in a wider variety of fields than wired networks. These low-power sensing nodes cannot perform high levels of computation for various applications, such as in industry. So, wired support sensing nodes are implemented for advanced sensing applications. These hybrid networks of wireless and wired nodes have been simulated under TinyOS. TinyOS seems to support energy-efficient communications in these hybrid networks [113].

F. TinyOS Implementation for Heterogeneous Networks

Sensing nodes are different from one another in terms of their energy levels, supported hardware platforms and sensing operations. Such heterogeneous network conditions can be produced with the help of the TiQ framework. TinyOS provides a TiQ environment for operation of heterogeneous networks. TinyOS appears to be the most reliable OS for heterogeneous networks [114].

TABLE IV
COMPARISON OF TinyOS-BASED SIMULATORS

Relevant References	Simulator	Purpose	Command line/GUI	Compatibility with Hardware Platforms	Supported Applications	Scalability	Bridging
[116], [117], [118], [119], [120], [121], [122], [123], [124], [125], [126]	TOSSIM	It simulates thousands of TinyOS-based nodes with bit granularity	Command line and graphical with TinyViz	All TinyOS supported platforms	CTP, routing algorithms	Yes	Yes
[127], [128], [129]	PowerTOSSIM	Estimating power consumption of sensing nodes	Graphical with help of TinyViz	MICA2	Predict power consumption of communications protocols	Yes	Yes
[130], [131], [132]	mTOSSIM	Estimating battery life time	GUI	TeloS and MICA2	Localization, coverage range study	Yes	No
[133], [134], [135]	Viptos	Simulating TinyOS-based heterogeneous networks	GUI	IRIS and Eyes	Multi-hop routing	Yes	Yes
[136], [137], [138]	QualNet	Modelling of energy consumption and clock drift	Command line	Skimmer and MICAz	Mobility-based sensing applications	Yes	No
[139]	TOSSF	Designed for modelling radio signal propagation and node mobility	Command line	Sky motes	Smart dust projects	Yes	No
[118]	Avrora	For AVR execution of instruction	Command line	TICC 2430	Heterogeneous networks	No	No
[140]	SmartSim	Provides comprehensive power usage reports	Graphical	MICA2 and MICAz	TinyOS 2.0 applications can be simulated	No	Yes
[141], [142]	EmTOS	Runs whole TinyOS applications as a single module	Command line	TinyNode	Simulation of heterogeneous networks	Yes	No

VII. SIMULATORS FOR TinyOS

Researchers can now simulate different sensing applications and OSs with virtual environments. Real hardware implementation of new applications in distributed sensor nodes is much more time-consuming and more expensive. Different virtual environments have been created to simulate different applications and OSs for WSNs [115]. TinyOS and its various applications can also be simulated on a wide variety of simulators. Instead of a TinyOS installation with real sensing nodes, various experiments can be performed by running it on simulators on a PC. A comparative view of different simulators is given in Table IV. Different simulators that help TinyOS are explained below.

A. TOSSIM

This is a widely accepted simulator for TinyOS and its various applications. It analyzes TinyOS at a very basic level. TOSSIM can find many bugs in TinyOS and its various applications. Large numbers of nodes running TinyOS can be simulated using it [116]–[118]. TOSSIM is used for many applications to test their operational behavior. Some of the widely used applications that utilize TOSSIM for testing are given below.

- Different routing algorithms are first simulated using TOSSIM. Multihop routing algorithms are usually simulated on TOSSIM [119].
- Distributed binary consensus algorithms are extensively used in WSNs for finding dead nodes [120]. TOSSIM is widely used by these algorithms to check their operation [121]. TOSSIM also provides support for various other algorithms [119], [177].
- Collection tree protocol (CTP) is an extensively used routing protocol for WSNs. It is an energy-efficient routing protocol for efficient data collection, processing and transmission of processed data. TOSSIM is used in simulations of CTP. CTP is highly compatible with this simulation environment [122].
- Hopfield neural networks simulation is performed in TOSSIM. TOSSIM, in this case, provides a parallel and distributed computation environment for simulation of neural networks [123].
- Heterogeneous and large-scale WSNs can be simulated with the help of TOSSIM. It is the only TinyOS-based

simulator that provides simulations of heterogeneous networks. For this purpose, a simple if and then control structure is proposed in TOSSIM [124].
- Time complexity, or the running time of applications, can be assessed with the help of TOSSIM. TinyOS-based applications can now be assessed for their execution time to avoid any bottlenecks in systems. For this purpose, the number of time periods is estimated and then the time complexity of the application is assessed [125].
- Different visualizers have been added in TOSSIM. These visualizers add not only graphical support to TOSSIM, but provide simulation of real-time applications. Having 3D visualizers for real-time applications makes it the simulator of choice for TinyOS real-time applications [126].

TOSSIM is considered the de facto simulator for TinyOS applications. Furthermore, TOSSIM has also undergone many developmental changes, such as adding new RF models. With these modifications, simulation of TinyOS on different hardware platforms has now become more and more easy [129].

B. Power TOSSIM

Energy-scarce WSNs with TinyOS can be simulated to predict accurate energy consumption in sensing nodes. Power TOSSIM is an extension of TOSSIM for predicting power demand for TinyOS applications [127], [128].

C. mTOSSIM

This is an advanced simulator for novel TinyOS supportive applications. In mTOSSIM, usually the battery life of sensing nodes in relation to TinyOS applications is predicted. Compared to PowerTOSSIM and TOSSIM, in mTOSSIM, the sensing environment is taken into account, such as indoor or outdoor. mTOSSIM employs an advanced radio model in its operation of predicting the battery lifetime of sensing motes [130].

D. Viptos

Viptos is a graphical simulator for TinyOS applications. Viptos is a combination of two strong simulating tools, namely, Ptolemy and TOSSIM. Viptos can simulate heterogeneous sensor nodes running TinyOS. The main contribution of Viptos is its ability to support graphical environments for simulation of TinyOS applications [133].

E. QualNet

Sensor nodes running TinyOS on the MICA2 hardware platform can be simulated using the QualNet simulator. This simulator provides accuracy and scalability when using the MICA2 hardware platform [136].

F. TOSSF

Simulator for Wireless Ad-Hoc Networks (SWAN) is a well-known simulator for wireless ad hoc networks. TinyOS scalable simulation framework (TOSSF) is an advanced version of SWAN for simulating TinyOS, to gain more accuracy and flexibility [139].

G. Avrora

AVR instructions can be simulated with the help of the Avrora simulator. This is a TinyOS-based simulator. Heterogeneous networks can also be simulated with the help of Avrora [118].

H. SmartSim

This is a TinyOS-supported simulator. SmartSim is more closely related to TOSSIM. However, the main difference lies in its graphical interface. SmartSim is a graphical simulator that is used for simulating TinyOS-based applications [140].

I. EmTOS

EmTOS [141] is based on EmStar [142]. The wrapper library of EmTOS, which is similar to that of TOSSIM, enables TinyOS applications to run a simulation as a single module. With the help of EmTOS, heterogeneous networks can also be simulated using TinyOS features.

VIII. Comparative View of TinyOS With Other Sensors OS

TinyOS is a widely used OS for sensing nodes. Randomly distributed sensing nodes use different hardware platforms. So, a wide variety of OSs have been developed for tiny sensing nodes [10]. Comparison of TinyOS with other OSs is based on their different characteristics and performance. This section discusses the comparative view of TinyOS based on its features and performance.

A. Comparative View of TinyOS Based on Its Features

A comparative view of TinyOS against Contiki, LiteOS, SOS, MANTIS, Nano-Rk, and RETOS OSs based on different features is given in Table V. Brief details of each OS are given as follows.

B. Contiki

The Contiki OS is discussed elsewhere [8], [10], [11], [20], [92], [143]–[145] as an open source OS for tiny sensing motes. Its multitasking kernel made it the OS for a wide variety of sensing motes. This lightweight, portable OS has traits of preemptive multithreading, proto-threads and virtual network computing. Contiki also provides support for a wide variety of communications protocols. Compared to TinyOS, this OS has dynamic and modular support for its different programming model components. The C language was used in designing the Contiki OS, as opposed to NesC, which was used in TinyOS. Now, the advanced version of Contiki 2.2.1 provides concurrency, ContikiSec for security and a Coffee file system. Event-driven characteristics of Contiki and TinyOS make them OSs of choice for sensing nodes. These enhanced features of Contiki made it more similar to TinyOS. In contrast to TinyOS, a managed memory allocator in Contiki provides more efficient memory management than TinyOS. Compared to TinyOS, Contiki OS has two types of events: asynchronous events and synchronous events.

TABLE V
COMPARATIVE VIEW BETWEEN TinyOS, AND OTHER WSN OSs

Features	TinyOS	Contiki	LiteOS	SOS	MANTIS	Nano-RK	RETOS
Relevant References	This study	[8], [10], [11], [20], [92], [143], [144], [145]	[8], [20], [146], [147]	[8], [11], [148]	[8], [10], [11], [149], [150]	[8], [11], [151]	[8], [152], [153]
Publication year	ASPLOS 2000	EmNets 2004	IPSN 2008	MobiSys 2005	MONET 2005	RTSS 2005	IPSN 2007
Static/Dynamic System	Static	Dynamic	Dynamic	Dynamic	Dynamic	Static	Dynamic
Monolithic or Modular System	Monolithic	Modular	Modular	Modular	Modular	Monolithic	Modular
Networking Support	Active message	ulP,ulP6,Rime	File-Assited	Message	”comm”	Socket	Three-layer architecture
Real-time Guarantee	No	No	No	No	No	Yes	Posix 1003.1b
Language Support	NesC	C	LiteC++	C	C	C	C
Event-Based Programming	Yes	Yes	Yes	Yes	Yes	No	No
Multi-Thread Support	Partial (through tiny threads)	Yes	Yes	No	Yes	Yes	Yes
Wireless Reprogramming	Yes	Yes	Yes	Yes(Modular)	No	No	Yes
File System	Single level(ELF, Matchbox)	Coffee	Hierarchical Unix-like	No	No	No	No
Remote Debugging	Yes (Clairvoyant)	No	Yes(DT)	No	Yes(Node MD)	No	No
Simulators	TOSSIM, Power TOSSIM, Viptos, Qualnet, TOSSF	Cooja, MSP-Sim Netsim	Through AVRORA	Java SOS	XMOS	AVR Studio 4	Not available
Communication Security	TinySec	ContikiSec	Not available	Not available	Not available	Not available	Not available
Hardware Platforms	MICA2, MICAz, TelosB/TMote Sky, Intel-Mote2, eyes, tinynode, IRIS, shimmer, TI CC2430 (testing)	ESB, TelosB/Tmote Sky	MICAz, IRIS	MICA2, MICAz, TelosB/Tmote Sky, XYZ	MICA2, MICAz, Telos, MANTIS nymph	MICAz, FireFly	MICAz, TelosB/TMote Sky, TI CC2430
Shell	Not available	Unix-like shell runs on sensor mote	Shell that runs on base station	Not available	Unix-like shell runs on sensor mote	Not available	Shell runs on mote(CH)
Resource Sharing	Virtualization and completion events	Serialized access	Through synchronization primitives	Serializesd access	Through semaphores	Serialized access through mutexes and semaphores	Virtualization
Energy Consumption	HPL, TOSSTI, EATT	Chameleon architecture, Micro-contrller is put in sleep mode	Smaller footprint saves energy	Virtual battery	Puts the scheduler in Sleep mode	Guaranteed and controlled access to CPU saves energy	Support through multi-hop networking
Concurrency	Yes	Yes	No	No	Yes	Yes	No
Supported Sensing Applications	Habitat monitoring, medical, industry, tracking, management	IP Net, Badgers	Unix-based applications	Navigation, Obstacle detection,	Fire net	Surveillance and environmental monitoring,	Motor controller, localization
Advance Versions	TinyOS 2.0	Contiki 2.2.1	LiteOS 1.0	SOS version 2.0.1	MANTIS 1.0 beta	Nano-Rk 1.0	RETOS 1.4

C. LiteOS

LiteOS [8], [20], [146], [147] is the Unix-based OS for sensing motes. This OS has gained much attention due to its ability to support Unix hardware platforms. LiteC++ with class library support is the programming language of this OS. Dynamic memory allocation and the modular component mode of LiteOS make it the OS for Unix-based hardware platforms. Wireless reprogramming capability, a built-in hierarchical Unix-based file system, and a smaller footprint are the distinguishing features of LiteOS. A new version, LiteOS 1.0, has been introduced to make it more responsive to real-time traffic. This new version, with the help of a virtual battery, conserves a lot of energy in systems. IRIS and MICAz are the well-known hardware platforms that

are supported by LiteOS. An event-based programming model, multi-thread support, and networking support make it closer to TinyOS, but compared to TinyOS, it does not support real-time traffic and does not provide concurrency.

D. SOS

SOS, discussed in several studies [8], [11], [148], was developed by Mobisys in 2005. The main motive behind its development was to introduce the OS to a WSN environment that can support multiple hardware platforms. SOS can support MICA2, MICAz, Telos and many other hardware platforms. Usually, Java-based simulators are employed for its testing before installation. Various differences exist between SOS and TinyOS. The main difference lies in the visibility or non-visibility of components during compilation. Components of TinyOS are not visible when they are compiled into binary code, whereas components of SOS do not disappear after compilation. Also, SOS does not provide a real-time guarantee, concurrency, multi-thread support, a compact file system and remote debugging. These features are well supported in TinyOS. The new version of SOS is 2.0.1. This advanced version supports a virtual battery and energy-efficient network message passing, and it has a modular structure. SOS uses dynamic memory management, just like TinyOS.

E. MANTIS

Event-driven TinyOS is quite different from the multi-threaded MANTIS OS. MANTIS is quite predictable and is used for a network that has to be idle for a long time [8], [10], [11], [149], [150]. MANTIS was developed under the MONET project of 2005, and the main purpose of its development was to ensure enhanced multithreading in a WSN OS. Binary semaphores and counting semaphores were introduced in MANTIS to ensure concurrency, just like in TinyOS. Compared to TinyOS, it supports dynamic memory allocation, a modular component model, and an event-based programming model. Wireless reprogramming, remote debugging, communications security and an improved file system are absent in the MANTIS OS. It also supports a wide variety of hardware platforms, just like TinyOS.

F. Nano-RK

In several studies [8], [11], [151], the main features of Nano-RK and its characteristic differences compared to TinyOS are discussed. It supports the time-sensitive applications of WSNs in more efficient ways, and Nano-RK was developed for handling of real-time tasks. A more sensitive and efficient task-scheduling technique has also been integrated into this OS. Resources are reserved in this OS to ensure timely and guaranteed delivery of network packets. Nano-RK has more similarity to TinyOS in that it has the same real-time handling of tasks, static memory management, monolithic system model, multi-thread support, and concurrency control. Compared to TinyOS, Nano-Rk does not have an efficient file system, remote debugging, and communications security. The current version of Nano-RK supports sockets, like abstractions for network communications.

TABLE VI

REDUCTION IN CODE DUE TO INLINING AND ITS EFFECT ON TinyOS PERFORMANCE (ADAPTED FROM [27])

Application	Inlined code size	Noninlined code size	code reduction	Data size	CPU reduction
Surge	14794	16984	12%	1188	15%
Mate	25040	27458	9%	1710	34%
TinyDB	64910	71724	10%	2894	30%

TABLE VII

REDUCTION IN CPU CYCLES, BOUNDARY CROSSING OF 7 MODULES, AND TIMER OVERHEAD DUE TO TinyOS CODE OPTIMIZATION (ADAPTED FROM [27])

Cycles	Optimized	Unoptimized	Reduction
Work	371	520	29%
Boundary crossing	109	258	57%
Non-interrupt	8	194	95%
Interrupt	101	64	-36%
Total	480	778	38%

G. RETOS

RETOS [8], [152], [153] was developed under a project of IPSN in 2007. Multi-threading for sensing motes is the main contribution of this OS. User-mode and kernel-mode handling of tasks is also incorporated in RETOS. RETOS supports the design and development of various sensing applications because it was used as a code checker for new sensing applications that run on it. It is now widely used in network communications due to the presence of a three-layer network architecture module. A wireless reprogramming capability, multi-threading, and dynamic memory allocation makes it more similar to TinyOS, whereas remote debugging, a compact file system and enhanced features for energy conservation are not addressed in RETOS. The latest version of RETOS is 1.4, which supports a wide variety of hardware platforms and virtualization.

H. Comparative View of TinyOS Based on Its Performance

TinyOS developers employ various techniques and design approaches in programming model of TinyOS to improve its performance. By adopting the various approaches and modifications in NesC code, a significant improvement in various performance metrics have been witnessed. TinyOS program size, random-access memory (RAM) usage, energy consumption, application code length, and central processing unit (CPU) utilization are the performance metrics that have been discussed in literature for performance comparison of TinyOS with other OSs [8]. The effects on performance metrics with respect to different coding paradigms have been discussed below.

1) Inlining in TinyOS Code: Inlining a function in NesC code as discussed in [28] improves the performance of TinyOS while minimizing various overheads such as code size, complex components, and wiring overheads. CPU utilization and energy consumption are minimized to a great extent by

TABLE VIII
PERFORMANCE IN ENERGY CONSERVATION ENHANCES DUE TO DYNAMIC UPDATES IN TinyOS (ADAPTED FROM [61])

Application	Component	Size (B)	Transfer energy	Deluge size	Saving factor
Blink	OS-Comp	6616	465.1	33726	5.1
	Blink	824	57.93	33726	40.9
	Leds	1728	121.48	33726	19.5
	Scheduler	1980	139.19	33726	17
Sense	OS-Comp	17040	1197.11	34074	2
	Sense	940	66.08	34074	36.25
	Leds	1728	121.48	34074	19.7
	Scheduler	2576	181.09	34074	13.2
Oscilloscope	OS-Comp	39328	2764.75	34504	0.87
	Oscilloscope	2008	141.16	34504	17.1
	Led	1720	120.91	34504	20.0
	Scheduler	3728	262.07	34504	9.25

inlining the code. Code is reduced by inlining a function with the help of single call site. Multiple call sites can extend the code which really affects the optimization of TinyOS. Table VI shows the performance of TinyOS with respect to CPU utilization when the inlining reduces the code of different applications such as surge, mate, and TinyDB.

With the help of inlining, sensing tasks are made smaller in size. Smaller sensing tasks are well supported in TinyOS compared to other OSs of WSNs. For example, MANTIS has 17.15% longer execution time for small task compared to TinyOS. This improved performance of TinyOS is also due to the efficient packet forwarding mechanism adopted by NesC [225].

2) Dynamic TinyOS: Sensing nodes are installed in a location where their replacements and their components update during runtime incur serious overhead. TinyOS updates and software exchanges cannot be done during sensing operations. For OS updates and component exchanges, sensing operations have to be shut down. This was a drawback of OSs for WSNs, including TinyOS. However, dynamic TinyOS feature in programming model of TinyOS lets developers update, and even exchange, the TinyOS components and software dynamically without interrupting the sensing operation. This turns out to be more efficient for memory, and a more energy-efficient approach for TinyOS applications [61]. The increase in performance of TinyOS (saving factor) with respect to energy while using TinyDB for various applications such as sense, blink, and oscilloscope is given in Table VIII.

When compared with OSs of WSNs, TinyOS shows more performance in conserving the energy of the network. For example, with the node at the position ($n = 8$) in tree and executing a sensing task of ($ls = 1ms$) size (where n represents the position of any sensing node in a binary tree, and ls is the duration for executing the task as defined in [225]), TinyOS shows 7.6% improvement in energy saving compared to MANTIS OS [225].

3) Incremental Programming of TinyOS Code: TinyOS code has been edited to provide the incremental network programming for WSNs. Unlike the dynamic programming as discussed in previous subsection, developers now use an algorithm named as Rsync in incremental programming. The sensing motes now can transmit their modified code during transmission without interrupting the sensing operation.

TABLE IX
TinyOS USES LESS MODULES AND LINES COMPARED TO APPLICATION MODULES AND LINES (ADAPTED FROM [27])

Application	Modules	OS Modules (% of full OS)	Lines	OS Lines (% of full OS)
Surge	31	27 (25%)	2860	2160 (14%)
Mate	35	28 (25%)	4736	2524 (17%)
TinyDB	65	38 (35%)	11681	4160 (28%)

With the use of Rsync algorithm, TinyOS shows 9.1% improvement in performance compared to LiteOS for changing the source code by implementing incremental programming in multi-hop WSNs [146], [226].

4) Optimization of NesC Code: Cross-components in NesC code are optimized by imposing the restrictions on component model to perform the static analysis. Call-graph for any application is constructed by NesC that excludes the unreachable code and hence improves the performance. This optimization of TinyOS code results in reduction in memory usage [27] and CPU cycles. Table VII shows the reduction in CPU cycles in the timer event, while optimizing the code of TinyOS. A total of 38% reduction in CPU cycles, 57% reduction in overhead of boundary crossing of 7 modules, and 29% reduction in overhead that is created by timer event can be achieved with optimization of TinyOS code.

5) Component Model of TinyOS Code: Which component of TinyOS code will handle which application, is decided by the component model of TinyOS code. Usually, the TinyOS code is divided into application code and the OS code. The OS code which consists of scheduler and radio stack handles the application code with the help of component model. Component model efficiently uses the code and assigns minimum OS code to execute each application. TinyOS source code contains 108 code modules and 64 modules for configuration. Every module has an average of 120 lines. It shows the efficient component model of TinyOS, which carries smaller code size. Table IX shows that the component model of TinyOS assigns minimum possible code to each of three applications run by TinyOS. This effective component model makes TinyOS not only the memory efficient but also the

TABLE X
TinyOS-BASED SUPPORTED HARDWARE PLATFORMS

Relevant References	Hardware Platform	CPU	Power	Memory	I/O Sensors	Radio	Remarks
horizontal line [8], [125], [157], [162], [165], [166]	MICA2	ATMEGA 128	.036mW sleep and 60mW active	4K RAM and 128K Flash	Large expansion connector	76Kbps	Primary TinyOS development platform
[8], [125], [157], [162]	MICAz	ATMEGA 128	.030mW sleep and 60mW active	4K RAM and 128K Flash	Large expansion connector	250Kbps	Supports IEEE 802.15.4 standard. Allows higher layer ZigBee standard
[8], [20], [157], [162]	TelosB/TMote	Motorola HCS08	.001mW sleep and 32mW active	4K RAM	USB and Ethernet	250Kbps	Supports IEEE 802.15.4 standard. Allows higher layer ZigBee standard. 1.8V operation
[162]	Rene	ATMEL 8535	.036mW sleep and 60mW active	512B RAM and 8K Flash	Large expansion connector	10Kbps	Primary TinyOS development platform
[8], [162]	Intel-Mote2	ARM 7TDMI 12- 48MHz	1mW idle and 120mW active	64KB SRAM and 512KB Flash	UART, USB, GPIO, 12C, SPI	Bluetooth 1.1	Multihop using scatternets, easy connections to PDAs, phones, TinyOS 1.0, 1.1.
[162]	BT Node	ATMEL Mega 128L 7.328Mhz	50MW idle and 285MW active	128KB Flash 4KB EEPROM and 4KB SRAM	Expandable connectors	Bluetooth	Cell phones, Supports TinyOS, multihop using multiple radios/nodes
[32], [162]	Stargate	Intel PXA255	.040mW sleep and 8x0mW active	64KNSRM	2PCMICA/CF, com ports, Ethernet, USB	Serial connection to sensor network	Flexible I/O and small form factor power management.

energy efficient, as less energy is consumed for execution of applications [27]. Performance comparison with respect to memory usage metric is made with MANTIS in [225]. An experimental application is executed on PIC16 [227] processor while using TinyOS and MANTIS. TinyOS comes out to be 4kB less in memory usage compared to MANTIS. This reduction in memory usage is due to the reduction in code performed by component model of TinyOS code.

IX. TinyOS SUPPORTED HARDWARE PLATFORMS AND THEIR FIELD IMPLEMENTATION

WSNs consisting of tiny motes perform three distinguishing tasks, compared to other conventional network nodes. These nodes are to sense, process and then transmit data to other connected nodes or to some other aggregating destination. These nodes work in a flat topology and, sometimes, in a clustered topology. These topological advances enable sensor nodes to conserve more and more energy [158]–[160]. Highly dense and resource-constrained networks of these sensing motes require hardware platforms that meet their requirements [161]. Design of the OS plays a crucial role in the sensing operation. TinyOS is considered to be more compatible with many hardware platforms that are designed for WSNs [162]. A variety of hardware platforms are supported by TinyOS. Different hardware platforms with their features are presented in Table X. To port from one hardware platform to another, TinyOS developers have come out with a hardware abstraction architecture. TinyOS hardware abstractions can be broadly classified into three layers [25], [163], [164]. These layers are as follows.

1) Hardware Interface Layer (HIL):
   This comprises hardware-independent components, interfaces and events.
2) Hardware Presentation Layer (HPL):
   This is close to the hardware layer. Components in this layer are not picked by applications but are used by hardware in some particular tasks.
3) Hardware Adaptation Layer (HAL):
   This layer favors hardware functionality, and is closer to the HPL.

Different layers within different hardware abstractions of hardware platforms are shown in Fig. 6 and Fig. 7. Hardware abstractions clearly show that certain applications are hardware-independent and others are hardware-dependent on different hardware platforms running TinyOS. Below are some hardware platforms that are supported by TinyOS.

- MICA is a platform that supports TinyOS. MICA, which is a very small hardware structure, usually in inches, can be used in multihop routing, and is a widely used platform [165]. Some other advanced versions of MICA are now deployed in health and other fields. These new releases of MICA enhance the use of TinyOS in different dimensions.
- MICA2-based platform extensions are available and use TinyOS [166]. MICA2 is the primary TinyOS design and

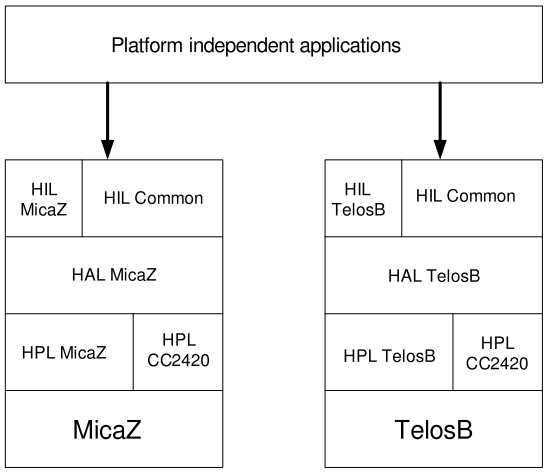

Fig. 6. Platform-independent applications (adapted from [163]).

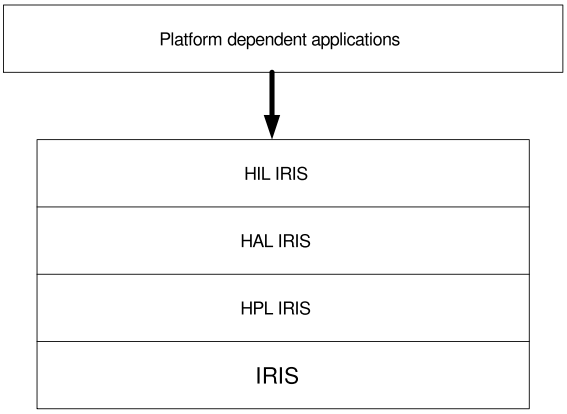

Fig. 7. Platform-dependent applications (adapted from [163]).

development platform. It supports almost all the TinyOS-supported sensing applications.

- MICAz [8], [125], [157], [162] hardware platforms enable TinyOS to support the IEEE 802.15.4 standard and other higher-layer ZigBee standards.
- Telos [8], [20], [157], [162] is the hardware platform that is designed for power conservation. Telos works well with TinyOS and saves power, which enhances network lifetime. It also supports the IEEE 802.15.4 standard. Usually, a 1.8V operation is supported in Telos.
- Rene [162] is the hardware-dependent hardware platform that runs TinyOS. Rene motes are generally employed in the medical sciences [168].
- Intel-Mote2 [8], [162] is the Linux-based TinyOS platform, which is very useful for many sensing applications. The main feature of this platform is that it can also be used for non-TinyOS-based sensing motes.
- BT Node [162] is the TinyOS-supported hardware platform that is especially designed for cellular phones supporting sensing applications that run TinyOS. Also, this hardware facilitates multihop communications.
- Stargate [32], [162] is the most energy-efficient hardware platform, newly developed for sensing motes. It supports serial connections to sensor networks.

TABLE XI
TinyOS AND OTHER OSs AND THEIR FIELD IMPLEMENTATIONS (ADAPTED FROM [20])

S.NO	Sensing Field	OS Used
1	Habitats	TinyOS
2	Minefields	Customized Linux
3	Battlefields	TinyOS
4	Lines in the sand	TinyOS
5	Counter-snipers	TinyOS
6	Electro-shepherds	Unknown
7	Virtual fences	Linux
8	Oil tankers	Unknown
9	Enemy vehicles	TinyOS
10	Trove games	TinyOS
11	Elder RFIDs	TinyOS
12	Murphy potatoes	TinyOS
13	Firewxnet	MANTIS OS
14	AlarmNet	TinyOS
15	Ecuador volcano	TinyOS
16	Pet games	TinyOS
17	Plugs	Custom
18	B-Live	Custom
19	Biomotion	No OS
20	AID-N	Other
21	Firefighting	TinyOS
22	Rehabil	TinyOS
23	CargoNet	Custom
24	Fence monitors	ScatterWeb
25	BikeNet	TinyOS
26	BriMon	TinyOS
27	IP net	Contiki
28	Smart homes	TinyOS
29	SVATS	TinyOS
30	Hitchhiker	TinyOS
31	Daily morning	TinyOS
32	Heritage	TinyOS
33	AC meter	TInyOS
34	Coal mines	TinyOS
35	ITS	Custom
36	Underwater	Custom
37	PipeProbe	Custom
38	Badgers	Contiki
39	Mount St. Helens volcano	TinyOS
40	Tunnels	TinyOS

A. TinyBench

In WSNs, there is no standardized approach for evaluating hardware platforms of sensing motes. Hardware platforms are selected based on their supported applications and other features. A standardized benchmark procedure is missing for categorizing hardware platforms. However, TinyOS developers have come out with a new TinyOS-based benchmark for hardware platforms of sensing motes. This benchmark is named TinyBench. This is a single-node standardized benchmark based on TinyOS applications [170].

Sensing nodes are employed in a variety of fields. Different OSs are used in different sensing mechanisms. Some OSs are more responsive to a particular environment where others do not perform well. TinyOS is employed in a variety of fields to enhance sensing operations. An overview of TinyOS versus other OSs with their implementations in different fields is given in Table XI [20]. Fig. 8 [20] shows an analysis of usage of OSs in WSNs. TinyOS is more widely deployed than any other OS. More than 60 percent of the sensing fields employ TinyOS due to its flexibility in the architecture.

TABLE XII
TinyOS SUPPORTED SENSING APPLICATIONS

No.	TinOS-Supported Sensing Applications	Relevant References
1	TinyOS for Water Monitoring Applications	[172], [173], [174], [175], [176]
2	TinyOS Support for Biological Applications	[166], [180], [181], [182], [183], [184]
3	Management Systems for Sensor Networks	[36], [63], [188], [189], [190], [191], [192]
4	TinyOS Supportive Applications for Detecting Security Threats in WSNs	[196]
5	TinyOS-based Environment Monitoring Applications	[200], [201], [202], [203]
6	TinyOS-based Agricultural Applications	[207]
7	TinyOS based Solar power generation monitoring applications	[208], [209]
8	TinyOS-based Habitat monitoring applications	[212], [213], [214]
9	TinyOS Implementations on FPGA Systems	[217], [218]
10	TinyOS Support for SANETs	[219]

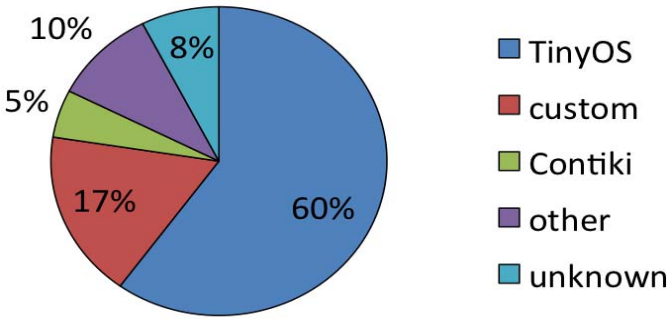

Fig. 8. Usage of different OSs (adapted from [20]).

X. TinyOS AND ITS SUPPORTED SENSING APPLICATIONS

TinyOS is an open OS for WSNs. Component-based TinyOS is a widely used OS, compared to other OSs in WSNs. TinyOS supports many sensing applications. Table XII shows the various TinyOS supported sensing applications and relevant references. Some of the widely used sensing applications that TinyOS supports are discussed below.

A. TinyOS for Water-Monitoring Applications

Water monitoring has gained a lot of attention in the sensing world. Water levels are measured to keep levels of water flow optimal. Water distribution networks are now monitored with the aid of sensing motes, which has resulted in increased efficiency of water distribution systems. Water waste is now kept to a minimum with the help of sensor nodes [171]. TinyOS supports many water monitoring applications. Water is remotely monitored with the help of TinyOS-supported applications that involve monitoring of both water quality and quantity. For remote water quantity monitoring, flooding routing protocols have been developed. Water quality sensors are employed to perform real-time monitoring for water contaminants [172]. Water quantity is measured with the help of smart water meters. These smart meters use ZigBee or M-Bus platforms. M-Bus implementations using TinyOS are very useful for water meter reading applications. TinyOS shows support for a wide variety of water-monitoring applications [173]. Underwater applications, such as seismic monitoring, are also very well supported by TinyOS. Underwater communications protocols, especially routing protocols, work well with TinyOS [174], [175]. Other communications networks, such as software-defined networks, also employ TinyOS for underwater communications [176].

B. TinyOS Support for Medical Applications

WSNs are now widely utilized in various medical applications. With the implementation of sensors in the medical field, WSNs have gained a lot of attention. Different medical applications are now provided by WSNs [178], [179]. TinyOS seems to be the perfect OS for different biological sensing applications. Neural interfaces with TinyOS help to capture nerve signals during an electroencephalogram (EEG) [180]. Brain neural signals are sensed by TinyOS-based sensing motes. There are applications that not only sense neural signals but also provide neural recording [181], [182]. A new NesC TinyOS model has been proposed for distributed and parallel computation of neural networks. This new model includes initialization of a neural network, and relaxation and convergence of neural computation [183]. Many of the TinyOS-based sensing motes use the MICAz platform for accurate measuring of signals [166]. TinyOS is also the OS of choice for electrocardiogram (ECG) monitoring. This specially designed TinyOS-based ECG monitor uses the 868 MHz ISM frequency band [184].

C. TinyOS Supported Management Systems for WSNs

Sensing nodes are equipped with limited resources. If the whole sensing and communications operation is not properly managed, then it will result in considerable degradation of the whole network. Network lifetime decreases, and earlier death of nodes happens in the network. So systematic management of the network is required for WSNs [185]–[187]. Different management methods in WSNs are employed to efficiently utilize network resources. DISON, supported by TinyOS, is a generic management system for sensor nodes. The management job is done by one of the nodes in the network, and then the job is taken over by another node after a certain time period. This saves the energy of nodes in the network [188]. TinyOS is an efficient OS that conserves power in the system by properly managing energy [36], [63], [189]. Limited resources of sensor nodes results in poor QoS. However, TinyOS provides better QoS management with limited network resources [190]. 6LoWPAN mobility management is handled very efficiently with the help of TinyOS. This open source OS manages the mobility of IP-based WSNs and enhances

throughput of the whole network. [191]. WSN management can also be achieved with implementation of the IEEE 1451 standard. The introduction of a transducer for efficient transmission of information under the IEEE 1451 standard, and further implementation of this standard in TinyOS, provides reliable network management for sensing motes [192].

D. TinyOS Supportive Applications for Detecting Security Threats in WSNs

Sensing nodes are randomly distributed. Location of these nodes is far away from the BS. Tiny motes sense data and then forward them to the BS, which is located at some distance. In this transmission, sensing node data can undergo various attacks. Data integrity and authenticity is lost when WSNs come under these attacks. Various classifications of attack have been discussed in the literature on sensing nodes [193]–[195]. Studies of different attacks have been performed on TinyOS. TinyOS minimizes threats of many attacks, and hence, it comes out to be a more secure and reliable OS for sensing nodes. Wireless injection attacks, denial of service attacks and man-in-the-middle attacks have been studied on TinyOS to minimize the effect of these attacks [196].

E. TinyOS-Based Environment Monitoring Applications

Environmental awareness has compelled researchers to come out with environmentally friendly tools. Environment degrading factors are monitored regularly. WSNs provide environment monitoring to save the environment from different degradation factors. Carbon emissions and glacier monitoring are tackled well by sensing motes [4], [197]–[199]. Many TinyOS-based environment monitoring applications have been developed to consistently monitor the environment. Greenhouse gases, glaciers, and global-warming monitoring applications are developed and tested with TinyOS. TinyOS, in these cases, comes out to be more efficient and reliable than any other OS for WSNs [200], [201]. WiseNet is a specially designed TinyOS-based wireless network for sensors. Wisenet is designed for monitoring the environment. Environmental factors such as light, temperature and humidity can be sensed well with TinyOS-based WiseNet. WiseNet maintains a database. Sensing nodes sense and forward data to servers, from where the data are taken for further analysis [202]. CC2430 sensing nodes implemented with TinyOS can also be used for measuring temperature and for monitoring switchgear assemblies. The switchgear assemblies encounter voltage changes. So TinyOS enables these nodes to detect voltage fluctuations and temperature very precisely [203].

F. TinyOS-Based Agricultural Applications

Now, WSNs have found their way into agriculture. Agricultural productivity can be increased many times over with the help of sensing nodes. The effect of weather on crops, of water levels on crops, the effects of fertilizers, and initial seed growth can be monitored with the help of sensing motes [204]–[206]. TinyOS-based applications have been developed that can be implemented on farms. TinyOS-based motes have been widely used for monitoring agricultural productivity [207].

G. TinyOS-Based Solar Power Generation Monitoring Applications

Renewable energy resources have gained a lot of attention in the modern energy-scarce world. These resources must be utilized with the utmost care to make them available for a longer period of time. Sensor networks are usually employed for monitoring these resources. Solar energy generation based on photovoltaic cells can also use TinyOS. TinyOS in these applications precisely monitors the whole generation process [208], [209].

H. TinyOS-Based Habitat Monitoring Applications

TinyOS is widely deployed in monitoring habitats. Habitat monitoring is an important application of WSNs. Through habitat monitoring, sensing nodes collect data related to residential areas and other biological habitats [210], [211]. TinyOS-based applications can be used for monitoring local harmonics. Through monitoring local harmonics, electricity issues can be resolved very efficiently [212]. Physical system monitoring applications, such as cyber physical system monitoring, have also gained attention. For this purpose, the IP flow information export (IPFIX) protocol was introduced for physical systems. This protocol has been implemented in TinyOS and was named TinyIPFIX. This is the provision of IoT applications, and TinyOS supports this combination of sensors and IoT implementation [213]. A more efficient TinyOS-based localization system was introduced [214]. This TinyOS-based application is an energy-efficient localization system, which uses an accelerometer. Nodes request the location of people, frequently and infrequently, depending upon whether people are running or standing still. Usually, TinyOS-based TelosB sensing-node platforms are employed in these operations.

I. TinyOS Implementations on FPGA Systems

A field programmable gate array (FPGA) is an integrated circuit that provides the customer with facility of configuration, so customers can configure these chips according to their own demands. FPGA systems are now used in WSNs [215], [216]. TinyOS is the only OS that has been made so flexible that it can easily be implemented on an FPGA system. Implementations of TinyOS on FPGA systems require few modifications in the code of TinyOS. TinyOS also seems to be very energy-efficient when ported to FPGA systems [217], [218].

*J. TinyOS Support for SANET*s

Sensor actuator networks (SANETs) have gained entry into many fields. SANETs are applied in agriculture, industry and in medical fields. SANETs can be employed for specific applications. For these purposes, TinyOS is adjusted to make it specific for that one field. TinyOS also supports Service-Oriented SANETs (SOSANETs). A new design of TinyOS, named TinySOA, is employed for this purpose [219].

XI. TinyOS Limitations and Modifications

TinyOS is a widely accepted OS for sensing nodes. However, it also faces many limitations. Certain limitations are discussed below.

A. Limitations of TinyOS

TinyOS is a widely used OS for WSNs. It has attracted many researchers and is being employed in many sensing motes. However, that does not mean that TinyOS supports all application tasks. There are certain limitations that TinyOS faces during the execution of tasks. Improper execution of tasks can lead to burdens and loads on the processor, which in turn, leads to inefficiency of the system [46]. Following are certain limitations faced by TinyOS.

1) Abnormal tasks in the job queue are not efficiently handled in TinyOS. Sometimes, abnormal tasks can hinder the execution of follow-up tasks. This can lead to extra burdens and loads on the processor [50].
2) TinyOS cannot handle a high frequency of local tasks. When the frequency of local tasks increases, it can lead to loss of other tasks.
3) Baud rate is highly affected when the execution of some tasks takes a lot of time, compared to other tasks. In this way, the execution of real tasks is affected. TinyOS cannot properly handle these situations [46].
4) TinyOS applications are considered difficult to construct, debug and handle [43].
5) While TinyOS performs well with static applications, TinyOS does not provide good performance for dynamic applications. Also, complex applications are not well supported in TinyOS [220].

   Most of the limitations are addressed with certain extensions and modifications in the programming structure of TinyOS. TinyOSs advanced versions and various extensions are given below.

B. TinyOS Advanced Versions and Extensions

With the growing demand for TinyOS, many new features were added. Many requirements for sensing nodes were not addressed in TinyOS when it was first developed. So users demanded that certain modifications and enhancements be incorporated.

1) TinyOS 2.0: The programming model of TinyOS 1.x was reconfigured and redesigned for TinyOS 2.0. Version 1.x encountered certain limitations. These limitations were addressed with enhanced features in 2.x, but this also introduced certain compatibility issues in TinyOS and its supported applications. TinyOS 2.0 advanced features are given below [221].

- Hardware abstractions were added in TinyOS 2.0. These hardware abstractions were named the hardware abstraction architecture, which can be further subdivided into three layers. This architecture enables TinyOS 2.0 to support a larger number of hardware platforms.
- In addition to a non-preemptive FIFO task scheduler in 1.x, TinyOS 2.0 has come up with a different scheduling approach. TinyOS enables programmers and developers to introduce a scheduler of their choice. Normally, in TinyOS 2.0, every task has its own reserved slot in the task queue.
- TinyOS 2.0 has an advanced and improved boot sequence. The StdControl interface of 1.x was partitioned into Init and StdControl interfaces in TinyOS 2.0. These two interfaces make the boot sequence more responsive by supporting start and stop commands.
- TinyOS 2.0 is written in the NesC 1.2 programming language. The programming model of TinyOS 2.0 is made in such a way that its components support virtualization in more reliable way.
- TinyOS 2.0 provides plenty of timer interfaces. Timers are an important feature for sensing motes. So, TinyOS 2.0 is considered more responsive, due to having plenty of timers.
- Provision of a message_t buffer in TinyOS 2.0 makes it more effective in sensing operations. There is plenty of space in this type of buffer, so it can handle a large number of packets.
- TinyOS 2.0 also provides efficient energy conservation methods. Usually, the power control mechanism of the microcontroller and the power control method of the device makes it more suitable for tiny sensing motes.

2) TinyWifi (Extension of TinyOS): TinyWifi is characterized as a Linux-based TinyOS. Sensor nodes and other sensing applications that are Linux-based can directly be implemented with the help of TinyWifi.This has saved developers time in further re-implementation of Linux-based sensing nodes. With the provision of TinyWifi, Linux-based PCs and other hand-held devices can be implemented on it. The primary purpose of TinyWifi is to use IEEE 802.11 communication protocols [163], [222]–[224].

XII. Conclusions

In this paper, we present the most widely used OS for WSNs, TinyOS. We have encompassed the main features of TinyOS. Contributions of this paper are multiple. This survey has shown not only the contemporary state of the art for TinyOS, but also the different developmental phases and revisions it has undergone. Its event-driven concurrency model, simple programming layout in NesC, and faster execution make it the OS of choice for tiny sensing nodes. We have shown that energy efficiency in TinyOS and the best scheduling algorithms, like real-time scheduling and priority scheduling, have made TinyOS operate the best in resource-constrained sensing environments. This paper has also pointed out that TinyOS code is very simple and short and takes less memory, compared to other conventional OSs. Memory management and protection with the help of TinyOS is simple and more novel. Less memory requirements for TinyOS installations and its applications has made it the OS for sensor nodes. TinyOS is so flexible that it provides support for the majority of energy-efficient routing protocols. Different simulators for TinyOS were also discussed in our research paper. A comparative view of TinyOS with other renowned

OSs for WSNs shows that TinyOS supports more features compared to other OSs for WSNs. TinyOSs support for a wide range of sensing applications, such as habitat monitoring and medical applications, has made it the OS of choice for WSNs. By going through the different features of TinyOS, it is clear that TinyOS is the OS for sensing networks that provides more accuracy and flexibility for sensing node applications to run.

REFERENCES

[1] A. Bharathidasan and V. A. S. Ponduru, "Sensor networks: An overview," Dept. Comput. Sci., Univ. California, Davis, CA, USA, Tech. Rep., 2002. [Online]. Available: http://www.csun.edu/~andrzej/COMP529-S05/papers/sensorNetworksSurvey.pdf

[2] C.-Y. Chong and S. P. Kumar, "Sensor networks: Evolution, opportunities, and challenges," *Proc. IEEE*, vol. 91, no. 8, pp. 1247–1256, Aug. 2003.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey of sensor network applications," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[4] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process. (ICASSP)*, vol. 4, May 2001, pp. 2033–2036.

[5] K. Martinez, J. K. Hart, and R. Ong, "Environmental sensor networks," *Computer*, vol. 37, no. 8, pp. 50–56, 2004.

[6] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Comput. Netw.*, vol. 38, no. 4, pp. 393–422, 2002.

[7] C. B. Margi *et al.*, "Impact of operating systems on wireless sensor networks (security) applications and testbeds," in *Proc. 19th Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2010, pp. 1–6.

[8] W. Dong, C. Chen, X. Liu, and J. Bu, "Providing OS support for wireless sensor networks: Challenges and approaches," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 4, pp. 519–530, Nov. 2010.

[9] P. A. Levis, "TinyOS: An open operating system for wireless sensor networks (invited seminar)," in *Proc. 7th Int. Conf. Mobile Data Manage. (MDM)*, 2006, p. 63.

[10] M. O. Farooq, S. Aziz, and A. B. Dogar, "Operating systems for wireless sensor networks: A survey," in *Future Generation Information Technology*. Berlin, Germany: Springer, 2010, pp. 616–631.

[11] A. M. V. Reddy, A. V. U. Phani Kumar, D. Janakiram, and G. A. Kumar, "Wireless sensor network operating systems: A survey," *Int. J. Sensor Netw.*, vol. 5, no. 4, pp. 236–255, 2009.

[12] P. Levis, "Experiences from a decade of TinyOS development," in *Proc. 10th USENIX Conf. OS Design Implement. (OSDI)*, 2012, pp. 207–220.

[13] L. Gu and J. A. Stankovic, "*t-kernel*: Providing reliable OS support to wireless sensor networks," in *Proc. 4th Int. Conf. Embedded Netw. Sensor Syst.*, 2006, pp. 1–14.

[14] R. Sugihara and R. K. Gupta, "Programming models for sensor networks: A survey," *ACM Trans. Sensor Netw.*, vol. 4, no. 2, 2008, Art. ID 8.

[15] M. Zheng, J. Sun, Y. Liu, J. S. Dong, and Y. Gu, "Towards a model checker for NesC and wireless sensor networks," in *Formal Methods and Software Engineering*. Berlin, Germany: Springer-Verlag, 2011, pp. 372–387.

[16] D. Bucur and M. Kwiatkowska, "On software verification for sensor nodes," *J. Syst. Softw.*, vol. 84, no. 10, pp. 1693–1707, 2011.

[17] S. Raman, "TinyOS—An operating system for tiny embedded networked sensors," presented at the Adv. Oper. Syst. Course, 2002.

[18] R. Gao, H. Zhou, and G. Su, "Structure of wireless sensors network based on TinyOS," in *Proc. Int. Conf. Control, Autom. Syst. Eng. (CASE)*, Jul. 2011, pp. 1–4.

[19] W. Archer, P. Levis, and J. Regehr, "Interface contracts for TinyOS," in *Proc. 6th Int. Conf. Inf. Process. Sensor Netw.*, 2007, pp. 158–165.

[20] G. Strazdins, A. Elsts, K. Nesenbergs, and L. Selavo, "Wireless sensor network operating system design rules based on real-world deployment survey," *J. Sens. Actuator Netw.*, vol. 2, no. 3, pp. 509–556, 2013.

[21] P. Levis *et al.*, "TinyOS: An operating system for sensor networks," in *Ambient Intelligence*. Berlin, Germany: Springer, 2005, pp. 115–148.

[22] L. Mottola and G. P. Picco, "Programming wireless sensor networks: Fundamental concepts and state of the art," *ACM Comput. Surv.*, vol. 43, no. 3, 2011, Art. ID 19.

[23] D. Gay, P. Levis, D. Culler, and E. Brewer. (2009). *nesC 1.3 Language Reference Manual*. [Online]. Available: http://nescc.sourceforge.ne

[24] V. Handziski, J. Polastre, J.-H. Hauer, and C. Sharp, "Flexible hardware abstraction of the TI MSP430 microcontroller in TinyOS," in *Proc. 2nd Int. Conf. Embedded Netw. Sensor Syst.*, 2004, pp. 277–278.

[25] V. Handziski, J. Polastre, J. Hauer, C. Sharp, A. Wolisz, and D. Culler, "Flexible hardware abstraction for wireless sensor networks," in *Proc. 2nd Eur. Workshop Wireless Sensor Netw.*, 2005, pp. 145–157.

[26] P. Levis. (2006). *TinyOS Programming*. [Online]. Available: http://www.tinyos.net/tinyos-2.x/doc/pdf/tinyos-programming.pdf

[27] D. Gay, P. Levis, R. Von Behren, M. Welsh, E. Brewer, and D. Culler, "The *nesC* language: A holistic approach to networked embedded systems," *ACM SIGPLAN Notices*, vol. 38, no. 5, pp. 1–11, 2003.

[28] D. Gay, P. Levis, and D. Culler, "Software design patterns for TinyOS," *ACM SIGPLAN Notices*, vol. 40, no. 7, pp. 40–49, 2005.

[29] P. Levis *et al.*, "The emergence of networking abstractions and techniques in TinyOS," in *Proc. NSDI*, 2004, p. 1.

[30] S. Iyengar, N. Parameshwaran, V. Phoha, N. Balakrishnan, and C. Okoye, "Tiny operating system (TinyOS)," in *Fundamentals of Sensor Network Programming: Applications and Technology*, vol. 1. New York, NY, USA: Wiley, 2011, pp. 92–97. DOI: 10.1002/9780470890158.ch5

[31] A. I. McInnes, "Using CSP to model and analyze TinyOS applications," in *Proc. 16th Annu. IEEE Int. Conf. Workshop Eng. Comput. Based Syst. (ECBS)*, Apr. 2009, pp. 79–88.

[32] K. Klues *et al.*, "TOSThreads: Thread-safe and non-invasive preemption in TinyOS," in *Proc. SenSys*, 2009, pp. 127–140.

[33] D. Bucur and M. Kwiatkowska, "Towards software verification for TinyOS applications," in *Proc. 9th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, 2009, pp. 400–401.

[34] P. Lindgren, H. Makitaavola, J. Eriksson, and J. Eliasson, "Leveraging TinyOS for integration in process automation and control systems," in *Proc. 38th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, 2012, pp. 5779–5785.

[35] M. Welsh and G. Mainland, "Programming sensor networks using abstract regions," in *Proc. NSDI*, 2004, p. 3.

[36] K. Klues *et al.*, "Integrating concurrency control and energy management in device drivers," in *ACM SIGOPS OS Rev.*, vol. 41, no. 6, pp. 251–264, 2007.

[37] J. Hill and D. Culler, "A wireless embedded sensor architecture for system-level optimization," UC Berkeley, Berkeley, CA, USA, Tech. Rep., 2002. [Online]. Available: http://www.cs.berkeley.edu/~culler/cs252-s02/papers/MICA_ARCH.pdf

[38] J. Hill, "System architecture for wireless sensor networks," Ph.D. dissertation, Dept. Comput. Sci., Univ. California, Berkeley, Berkeley, CA, USA, 2003.

[39] P. Levis. (2002). *TinyOS: Getting Started*. [Online]. Available: http://www3.nd.edu/~nest/benders_nest/doc/tos-developer.pdf

[40] M. H. Alizai, O. Landsiedel, and K. Wehrle, "Modeling execution time and energy consumption in sensor node simulation," *PIK-Praxis Informationsverarbeitung Kommunikation*, vol. 32, no. 2, pp. 127–132, 2009.

[41] P. Levis, D. Gay, and D. Culler, "Active sensor networks," in *Proc. 2nd Conf. Symp. Netw. Syst. Design Implement.*, vol. 2, 2005, pp. 343–356.

[42] N. Kothari, T. Millstein, and R. Govindan, "Deriving state machines from TinyOS programs using symbolic execution," in *Proc. Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Apr. 2008, pp. 271–282.

[43] A. R. Dalton, S. K. Wahba, S. Dandamudi, and J. O. Hallstrom, "Visualizing the runtime behavior of embedded network systems: A toolkit for TinyOS," *Sci. Comput. Program.*, vol. 74, no. 7, pp. 446–469, 2009.

[44] V. Borges, O. Raikar, V. Desai, and P. Dalvi, "A comparative study of TinyOS scheduling strategies and future scope," in *Proc. Int. Conf. Comput. Sci. Eng.*, Apr. 2012, pp. 83–87.

[45] P. Levis and C. Sharp, *Schedulers and Tasks*, document TEP 106, 2011.

[46] V. Subramonian, H.-M. Huang, S. Datar, and C. Lu, "Priority scheduling in TinyOS—A case study," Dept. Comput. Sci., Washington Univ., St. Louis, MO, USA, Tech. Rep. WUCSE-2003-74, 2003.

[47] T. Lei, X.-M. Zhao, and F. Hui, "A TinyOS scheduling strategy and its implementation," in *Proc. IEEE 3rd Int. Conf. Commun. Softw. Netw.*, May 2011, pp. 216–219.

[48] M. Yu, S. J. Xiahou, and X. Y. Li, "A survey of studying on task scheduling mechanism for TinyOS," in *Proc. 4th Int. Conf. Wireless Commun., Netw. Mobile Comput. (WiCOM)*, Oct. 2008, pp. 1–4.

[49] K. Atefi, M. Sadeghi, and A. Atefi, "Real-time scheduling strategy for wireless sensor networks O.S," *Int. J. Distrib. Parallel Syst.* vol. 2, no. 6, pp. 63–78, 2011.

[50] Y. Zhao, Q. Wang, W. Wang, D. Jiang, and Y. Liu, "Research on the priority-based soft real-time task scheduling in TinyOS," in *Proc. Int. Conf. Inf. Technol. Comput. Sci. (ITCS)*, vol. 1, Jul. 2009, pp. 562–565.

[51] S. Tak, H. Kim, and T. Kim, "A study on real-time scheduling for low-power sensor node platforms," in *Proc. IEEE 12th Int. Conf. Comput. Inf. Technol. (CIT)*, Oct. 2012, pp. 169–176.

[52] M. Khezri, M. A. Sarram, and F. Adibniya, "Simplifying concurrent programming of networked embedded systems," in *Proc. Int. Symp. Parallel Distrib. Process. Appl. (ISPA)*, Dec. 2008, pp. 993–998.

[53] M. Chen, V. C. M. Leung, S. Mao, and M. Li, "Cross-layer and path priority scheduling based real-time video communications over wireless sensor networks," in *Proc. IEEE Veh. Technol. Conf. (VTC Spring)*, May 2008, pp. 2873–2877.

[54] B. Hohlt and E. Brewer, "Network power scheduling for TinyOS applications," in *Distributed Computing in Sensor Systems*. Berlin, Germany: Springer-Verlag, 2006, pp. 443–462.

[55] J. L. Hill, "Electronic access control, tracking and paging system," U.S. Patent 7 367 497, May 6, 2008.

[56] T. Alliance, "TinyOS 2.1: Adding threads and memory protection to TinyOS," in *Proc. 6th ACM Conf. Embedded Netw. Sensor Syst.*, 2008, pp. 413–414.

[57] N. Cooprider, W. Archer, E. Eide, D. Gay, and J. Regehr, "Efficient memory safety for TinyOS," in *Proc. 5th Int. Conf. Embedded Netw. Sensor Syst.*, 2007, pp. 205–218.

[58] J. Regehr, N. Cooprider, W. Archer, and E. Eide, " Memory safety and untrusted extensions for TinyOS," School Comput., Univ. Utah, Salt Lake City, UT, USA, Tech. Rep. UUCS–06–007, 2006.

[59] A. Dunkels, O. Schmidt, T. Voigt, and M. Ali, "Protothreads: Simplifying event-driven programming of memory-constrained embedded systems," in *Proc. 4th Int. Conf. Embedded Netw. Sensor Syst.*, 2006, pp. 29–42.

[60] W. P. McCartney and N. Sridhar, "Stackless preemptive multi-threading for TinyOS," in *Proc. Int. Conf. Distrib. Comput. Sensor Syst. Workshops (DCOSS)*, 2011, pp. 1–8.

[61] W. Munawar, M. H. Alizai, O. Landsiedel, and K. Wehrle, "Dynamic TinyOS: Modular and transparent incremental code-updates for sensor networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2010, pp. 1–6.

[62] R. Min *et al.*, "Low-power wireless sensor networks," in *Proc. 14th Int. Conf. VLSI Design*, 2001, pp. 205–210.

[63] A. Sinha and A. Chandrakasan, "Dynamic power management in wireless sensor networks," *IEEE Design Test Comput.*, vol. 18, no. 2, pp. 62–74, Mar./Apr. 2001.

[64] N. Akilandeswari, B. Santhi, and B. Baranidharan, "A survey on energy conservation techniques in wireless sensor networks," *J. Agricult. Biol. Sci.*, vol. 8, no. 4, pp. 265–269, 2013.

[65] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad Hoc Netw.*, vol. 7, no. 3, pp. 537–568, 2009.

[66] Z. D. Purvis and A. G. Dean, "TOSSTI: Saving time and energy in TinyOS with software thread integration," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2008, pp. 354–363.

[67] O. Landsiedel, K. Wehrle, and S. Götz, "Accurate prediction of power consumption in sensor networks," in *Proc. 2nd Workshop Embedded Netw. Sensors*, May 2005, pp. 37–44.

[68] M. N. Halgamuge, M. Zukerman, K. Ramamohanarao, and H. L. Vu, "An estimation of sensor energy consumption," *Prog. Electromagn. Res. B*, vol. 12, pp. 259–295, 2009.

[69] S.-F. Li, R. Sutton, and J. Rabaey, "Low power operating system for heterogeneous wireless communication system," in *Compilers and Operating Systems for Low Power*. New York, NY, USA: Springer, 2003, pp. 1–16.

[70] S. Abbate, M. Avvenuti, D. Cesarini, and A. Vecchio, "Estimation of energy consumption for TinyOS 2.x-based applications," *Proc. Comput. Sci.*, vol. 10, pp. 1166–1171, Dec. 2012.

[71] S. Abbate, M. Avvenuti, A. Biondi, and A. Vecchio, "Estimation of energy consumption in wireless sensor networks using TinyOS 2.x," in *Proc. IEEE Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2011, pp. 842–843.

[72] S. K. Sarna and M. Zaveri, "EATT: Energy aware target tracking for wireless sensor networks using TinyOS," in *Proc. 3rd IEEE Int. Conf. Comput. Sci. Inf. Technol. (ICCSIT)*, vol. 1, Jul. 2010, pp. 187–191.

[73] B. Dezfouli, M. Radi, S. A. Razak, T. Hwee-Pink, and K. A. Bakar, "Modeling low-power wireless communications," *J. Netw. Comput. Appl.* vol. 51, pp. 102–126, May 2014.

[74] B. Sharma and T. C. Aseri, "A hybrid and dynamic reliable transport protocol for wireless sensor networks," *Comput. Elect. Eng.*, vol. 48, pp. 298–311, Nov. 2015.

[75] S. Shekhar, R. Mishra, R. K. Ghosh, and R. K. Shyamasundar, "Post-order based routing & transport protocol for wireless sensor networks," *Pervasive Mobile Comput.*, vol. 11, pp. 229–243, Apr. 2014.

[76] H. S. Kamath, "Energy efficient routing protocol for wireless sensor networks," *Int. J. Adv. Comput. Res.*, vol. 3, no. 10, pp. 95–100, Jun. 2013.

[77] N. Pantazis, S. A. Nikolidakis, and D. D. Vergados, "Energy-efficient routing protocols in wireless sensor networks: A survey," *IEEE Commun. Surveys Tutorials*, vol. 15, no. 2, pp. 551–591, 2013.

[78] A. Balamurugan, "An energy efficient fitness based routing protocol in wireless sensor networks," *ICTACT J. Commun. Technol.*, vol. 5, no. 1, pp. 894–899, 2014.

[79] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Netw.*, vol. 3, no. 3, pp. 325–349, 2005.

[80] H. Zhu and M. Li, "Opportunistic routing protocols," in *Studies on Urban Vehicular Ad-Hoc Networks*. New York, NY, USA: Springer, 2013, pp. 41–74.

[81] S. Biswas and R. Morris, "Opportunistic routing in multi-hop wireless networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 69–74, 2004.

[82] J. Carnley, B. Sun, and S. K. Makki, "TORP: TinyOS opportunistic routing protocol for wireless sensor networks," in *Proc. IEEE Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2011, pp. 111–115.

[83] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. IEEE 33rd Annu. Hawaii Int. Conf. Syst. Sci.*, Jan. 2000, pp. 1–10.

[84] C. Hou, K. W. Tang, and E. Noel, "Implementation and analysis of the LEACH protocol on the TinyOS platform," in *Proc. IEEE Int. Conf. ICT Converg. (ICTC)*, Oct. 2013, pp. 918–923.

[85] Y. Liang, "Study of protocol for wireless sensor network based on TinyOS," in *Proc. IEEE Int. Conf. Comput. Design Appl. (ICCDA)*, vol. 2, Jun. 2010, pp. V2-602–V2-605.

[86] S. Khan, S. Basharat, M. S. H. Khiyal, and S. A. Khan, "Investigating energy consumption of localized and non localized ad hoc routing protocols in TinyOS," in *Proc. IEEE Multitopic Conf. (INMIC)*, Dec. 2006, pp. 355–358.

[87] K. Daabaj, "Load-balanced routing scheme for TinyOS-based wireless sensor networks," in *Proc. IEEE Int. Conf. Wireless Inf. Technol. Syst. (ICWITS)*, Aug./Sep. 2010, pp. 1–4.

[88] W. Wang, J.-H. Youn, and H. R. Sharif, "The implementation of an energy balanced routing protocol with dynamic power scaling in TinyOS," in *Proc. IEEE Int. Conf. Sensor Netw., Ubiquitous, Trustworthy Comput.*, vol. 2, Jun. 2006, pp. 262–267.

[89] Z. Rehena, K. Kumar, S. Roy, and N. Mukherjee, "SPIN implementation in TinyOS environment using nesC," in *Proc. Int. Conf. Comput. Commun. Netw. Technol. (ICCCNT)*, Jul. 2010, pp. 1–6.

[90] M. Bertocco, G. Gamba, A. Sona, and S. Vitturi, "Performance measurements of CSMA/CA-based wireless sensor networks for industrial applications," in *Proc. IEEE Instrum. Meas. Technol. Conf. (IMTC)*, pp. 1–6, May 2007.

[91] I. Demirkol, C. Ersoy, and F. Alagoz, "MAC protocols for wireless sensor networks: A survey," *IEEE Commun. Mag.*, vol. 44, no. 4, pp. 115–121, Apr. 2006.

[92] J. Ko, N. Tsiftes, A. Dunkels, and A. Terzis, "Pragmatic low-power interoperability: ContikiMAC vs TinyOS LPL," in *Proc. 9th Annu. IEEE Commun. Soc. Conf. Sensor, Mesh Ad Hoc Commun. Netw. (SECON)*, Jun. 2012, pp. 94–96.

[93] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proc. 4th Int. Conf. Embedded Netw. Sensor Syst.*, 2006, pp. 307–320.

[94] D. van den Akker and C. Blondia, "MultiMAC: A multiple MAC network stack architecture for TinyOS," in *Proc. IEEE 21st Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul./Aug. 2012, pp. 1–5.

[95] P. Levis. (2007). *Packet Protocols*, TEP Core Working Group. [Online]. Available: http://www.tinyos.net/tinyos-2.x/doc/html/tep116.html

[96] S. Madden, J. Hellerstein, and W. Hong, "TinyDB: In-network query processing in tinyos," Intel Res., Tech. Rep. IRB-TR-02-014, Oct. 2002.

[97] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: An acquisitional query processing system for sensor networks," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 122–173, 2005.

[98] P. Di Felice, M. Ianni, and L. Pomante, "A spatial extension of TinyDB for wireless sensor networks," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2008, pp. 1076–1082.

[99] K. Mayer, K. Taylor, and A. N. U. Campus, "TinyDB by remote," in *Proc. World Conf. Integr. Design Process Tech.*, Austin, TX, USA, 2003, pp. 3–6.

[100] A. Manjeshwar and D. P. Agrawal, "APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks," in *Proc. 16th Int. Parallel Distrib. Process. Symp.*, vol. 2, 2002, p. 0195b.

[101] M. M. Afsar, and M.-H. Tayarani-N, "Clustering in sensor networks: A literature survey," *J. Netw. Comput. Appl.*, vol. 46, pp. 198–226, Nov. 2014.

[102] J. J. P. C. Rodrigues and P. A. C. S. Neves, "A survey on IP-based wireless sensor network solutions," *Int. J. Commun. Syst.*, vol. 23, no. 8, pp. 963–981, 2010.

[103] S. S. Bhunia, D. K. Sikder, S. Roy, and N. Mukherjee, "A comparative study on routing schemes of IP based wireless sensor network," in *Proc. 9th Int. Conf. Wireless Opt. Commun. Netw. (WOCN)*, Sep. 2012, pp. 1–5.

[104] J. Ko, S. Dawson-Haggerty, O. Gnawali, D. Culler, and A. Terzis, "Evaluating the performance of RPL and 6LoWPAN in TinyOS," in *Proc. Workshop Extending Internet Low Power Lossy Netw. (IP+SN)*, 2011, pp. 1–6.

[105] L. B. Saad, C. Chauvenet, and B. Tourancheau, "IPv6 (Internet protocol version 6) heterogeneous networking infrastructure for energy efficient building," *Energy*, vol. 44, no. 1, pp. 447–457, 2012.

[106] V. Kumar, G. Oikonomou, T. Tryfonas, D. Page, and I. Phillips, "Digital investigations for IPv6-based wireless sensor networks," *Digit. Investigation*, vol. 11, pp. S66–S75, Aug. 2014.

[107] A. Cunha, A. Koubaa, R. Severino, and M. Alves, "Open-ZB: An open-source implementation of the IEEE 802.15.4/ZigBee protocol stack on TinyOS," in *Proc. IEEE Int. Conf. Mobile Adhoc Sensor Syst. (MASS)*, Oct. 2007, pp. 1–12.

[108] P. Baronti, P. Pillai, V. W. C. Chook, S. Chessa, A. Gotta, and Y. F. Hu, "Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards," *Comput. Commun.*, vol. 30, no. 7, pp. 1655–1695, 2007.

[109] S. C. Ergen. (Sep. 10, 2004). *ZigBee/IEEE 802.15.4 Summary*, UC Berkeley. [Online]. Available: http://www.eecs.berkeley.edu/csinem/academic/publications/zigbee.pdf

[110] A. Clemotte, E. A. Vargas, and S. L. Toral, "A Zigbee target system running TinyOS," in *Proc. 15th Int. Power Electron. Motion Control Conf. (EPE/PEMC)*, Sep. 2012, pp. LS4e.5-1–LS4e.5-5.

[111] M. M. R. Mozumdar, F. Gregoretti, L. Lavagno, and L. Vanzago, "Porting application between wireless sensor network software platforms: TinyOS, MANTIS and ZigBee," in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2008, pp. 1145–1148.

[112] R. Sauter, O. Saukh, O. Frietsch, and P. J. Marrón, "TinyLTS: Efficient network-wide logging and tracing system for TinyOS," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 2033–2041.

[113] A. Anastasopoulos, D. Tsitsipis, S. Giannoulis, and S. Koubias, "Implementation and evaluation of a hybrid network utilizing TinyOS-based systems and Ethernet," in *Proc. IEEE Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2007, pp. 441–447.

[114] Y.-T. Wang and R. Bagrodia, "Scalable emulation of TinyOS applications in heterogeneous network scenarios," in *Proc. IEEE 6th Int. Conf. Mobile Adhoc Sensor Syst. (MASS)*, Oct. 2009, pp. 140–149.

[115] B. Musznicki and P. Zwierzykowski, "Survey of simulators for wireless sensor networks," *Int. J. Grid Distrib. Comput.*, vol. 5, no. 3, pp. 23–50, 2012.

[116] P. Levis and N. Lee, "TOSSIM: A simulator for TinyOS networks,"Comput. Sci. Division, Univ. California Berkeley, Berkeley, CA, USA, Tech. Rep., 2003, vol. 17.

[117] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," in *Proc. 1st Int. Conf. Embedded Netw. Sensor Syst.*, 2003, pp. 126–137.

[118] L. Zhao, C. Xiao-Yan, C. Meng-Xiao, and Z. Wei, "The design and implement of automated transfer based on TinyOS," in *Proc. 3rd IEEE Int. Symp. Microw., Antenna, Propag. EMC Technol. Wireless Commun.*, Oct. 2009, pp. 748–750.

[119] S. A. Notani, "Performance simulation of multihop routing algorithms for ad-hoc wireless sensor networks using TOSSIM," in *Proc. 10th Int. Conf. Adv. Commun. Technol. (ICACT)*, vol. 1. Feb. 2008, pp. 508–513.

[120] S. Kar and J. M. F. Moura, "Distributed consensus algorithms in sensor networks: Quantized data and random link failures," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1383–1400, Mar. 2010.

[121] A. Abdaoui and T. M. El-Fouly, "TOSSIM and distributed binary consensus algorithm in wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 41, pp. 451–458, May 2014.

[122] R. Nath, "A TOSSIM based implementation and analysis of collection tree protocol in wireless sensor networks," in *Proc. IEEE Int. Conf. Commun. Signal Process. (ICCSP)*, Apr. 2013, pp. 484–488.

[123] J. Li and G. Serpen, "TOSSIM simulation of wireless sensor network serving as hardware platform for Hopfield neural net configured for max independent set," *Procedia Comput. Sci.*, vol. 6, pp. 408–412, Dec. 2011.

[124] J. Li and G. Serpen, "Simulating heterogeneous and larger-scale wireless sensor networks with TOSSIM TinyOS emulator," *Procedia Comput. Sci.*, vol. 12, pp. 374–379, Dec. 2012.

[125] G. Serpen and J. Li, "Assessing time complexity of applications for TinyOS-Mica wireless sensor networks in TOSSIM emulator," *Procedia Comput. Sci.*, vol. 12, pp. 380–385, Dec. 2012.

[126] A. Gupta and S. Roy, "Design and implementation of visualizers for TinyOS," *Procedia Technol.*, vol. 10, pp. 409–416, Dec. 2013.

[127] V. Shnayder, M. Hempstead, B. Chen, and H. M. Welsh, "Powertossim: Efficient power simulation for TinyOS applications," in *Proc. ACM SenSys Sensor Netw.*, Los Angeles, CA, USA, 2003.

[128] M. Safaei, A. S. H. Ismail, and A. S. H. Ismail, "Visualization, data analyzing and energy usage analysis in wireless sensor network based on TinyOs and PowerTossimZ," *Int. J. Comput. Commun. Netw.*, vol. 1, no. 1, 2011.

[129] C. Suh, J.-E. Joung, and Y.-B. Ko, "New RF models of the TinyOS simulator for IEEE 802.15.4 standard," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2007, pp. 2236–2240.

[130] J. M. Mora-Merchan, D. F. Larios, J. Barbancho, F. J. Molina, J. L. Sevillano, and C. León, "mTOSSIM: A simulator that estimates battery lifetime in wireless sensor networks," *Simul. Model. Pract. Theory*, vol. 31, pp. 39–51, Feb. 2013.

[131] W. Dron, S. Duquennoy, T. Voigt, K. Hachicha, and P. Garda, "An emulation-based method for lifetime estimation of wireless sensor networks," in *Proc. IEEE Int. Conf. Distrib. Comput. Sensor Syst. (DCOSS)*, May 2014, pp. 241–248.

[132] A. Pughat and V. Sharma, "A review on stochastic approach for dynamic power management in wireless sensor networks," *Human-Centric Comput. Inf. Sci.*, vol. 5, no. 1, pp. 1–14, 2015.

[133] E. Cheong, E. A. Lee, and Y. Zhao, "Viptos: A graphical development and simulation environment for TinyOS-based wireless sensor networks," in *Proc. SenSys*, vol. 5, 2005, p. 302.

[134] G. Teng, K. Zheng, and W. Dong, "A survey of available tools for developing wireless sensor networks," in *Proc. IEEE 3rd Int. Conf. Syst. Netw. Commun. (ICSNC)*, Oct. 2008, pp. 139–144.

[135] H. Taylor, "Multihop routing simulation of TinyOS-based wireless sensor networks in Viptos," Dept. Elect. Comput. Eng., Univ. Vermont, Burlington, VT, USA, Tech. Rep., 2006. [Online]. Available: https://chess.eecs.berkeley.edu/superb/projects/taylor_paper_viptos.pdf

[136] M. Varshney, D. Xu, M. Srivastava, and R. Bagrodia, "sQualNet: A scalable simulation and emulation environment for sensor networks," in *Proc. Int. Conf. Inf. Process. Sensor Netw.*, New York, NY, USA, 2007, p. 24.

[137] C. P. Singh, O. P. Vyas, and M. K. Tiwari, "A survey of simulation in sensor networks," in *Proc. IEEE Int. Conf. Comput. Intell. Modelling Control Autom.*, Dec. 2008, pp. 867–872.

[138] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient, collision-free medium access control for wireless sensor networks," *Wireless Netw.*, vol. 12, no. 1, pp. 63–78, 2006.

[139] L. F. Perrone and D. M. Nicol, "A scalable simulator for TinyOS applications," in *Proc. Winter IEEE Simulation Conf.*, vol. 1, Dec. 2002, pp. 679–687.

[140] M. Safaei and A. S. H. Ismail, "SmartSim: Graphical sensor network simulation based on TinyOS and TOSSIM," in *Proc. IEEE 3rd Int. Conf. Intell. Syst., Modelling Simulation (ISMS)*, Feb. 2012, pp. 611–615.

[141] L. Girod *et al.*, "A system for simulation, emulation, and deployment of heterogeneous sensor networks," in *Proc. 2nd Int. Conf. Embedded Netw. Sensor Syst.*, 2004, pp. 201–213.

[142] L. G. J. E. A. Cerpa and T. S. N. R. D. Estrin, "EmStar: A software environment for developing and deploying wireless sensor networks," in *Proc. USENIX Annu. Tech. Conf., General Track*, 2004, pp. 283–296.

[143] T. Reusing, "Comparison of operating systems tinyos and contiki," *Sens. Nodes-Oper., Netw. Appl.*, vol. 7, 2012. DOI: 10.2313/NET-2012-08-2_02

[144] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki—A lightweight and flexible operating system for tiny networked sensors," in *Proc. 29th Annu. IEEE Int. Conf. Local Comput. Netw.*, Nov. 2004, pp. 455–462.

[145] G. Oikonomou and I. Phillips, "Experiences from porting the Contiki operating system to a popular hardware platform," in *Proc. IEEE Int. Conf. Distrib. Comput. Sensor Syst. Workshops (DCOSS)*, Jun. 2011, pp. 1–6.

[146] Q. Cao, T. Abdelzaher, J. Stankovic, and T. He, "The liteOS operating system: Towards Unix-like abstractions for wireless sensor networks," in *Proc. IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Apr. 2008, pp. 233–244.

[147] Q. Cao and T. F. Abdelzaher, "LiteOS: A lightweight operating system for C++ software development in sensor networks," in *Proc. IEEE 4th Int. Conf. Embedded Netw. Sensor Syst.*, Oct./Nov. 2006, pp. 361–362.

[148] C.-C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava, "A dynamic operating system for sensor nodes," in *Proc. ACM MobiSys*, 2005, pp. 163–176.

[149] S. Bhatti *et al.*, "MANTIS OS: An embedded multithreaded operating system for wireless micro sensor platforms," *Mobile Netw. Appl.*, vol. 10, no. 4, pp. 563–579, 2005.

[150] H. Abrach *et al.*, "MANTIS: System support for multimodal networks of in-situ sensors," in *Proc. 2nd ACM Int. Conf. Wireless Sensor Netw. Appl.*, 2003, pp. 50–59.

[151] A. Eswaran, A. Rowe, and R. Rajkumar, "Nano-RK: An energy-aware resource-centric RTOS for sensor networks," in *Proc. IEEE RTSS*, Dec. 2005, pp. 256–265.

[152] H. Cha *et al.*, "RETOS: Resilient, expandable, and threaded operating system for wireless sensor networks," in *Proc. ACM/IEEE IPSN*, Apr. 2007, pp. 148–157.

[153] H. Cha *et al.*, "The RETOS operating system: Kernel, tools and applications," in *Proc. 6th Int. Conf. Inf. Process. Sensor Netw.*, 2007, pp. 559–560.

[154] C. Duffy, U. Roedig, J. Herbert, and C. Sreenan, "An experimental comparison of event driven and multi-threaded sensor node operating systems," in *Proc. 5th Annu. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2007, pp. 267–271.

[155] S. Park, J. W. Kim, K.-Y. Shin, and D. Kim, "A nano operating system for wireless sensor networks," in *Proc. 8th Int. Conf. Adv. Commun. Technol. (ICACT)*, vol. 1, Feb. 2006, pp. 1–4.

[156] T. V. Chien, H. N. Chan, and T. N. Huu, "A comparative study on operating system for wireless sensor networks," in *Proc. IEEE Int. Conf. Adv. Comput. Sci. Inf. Syst. (ICACSIS)*, Dec. 2011, pp. 73–78.

[157] M. O. Farooq and T. Kunz, "Operating systems for wireless sensor networks: A survey," *Sensors*, vol. 11, no. 6, pp. 5900–5930, 2011.

[158] D. Kumar, T. C. Aseri, and R. B. Patel, "EEHC: Energy efficient heterogeneous clustered scheme for wireless sensor networks," *Comput. Commun.*, vol. 32, no. 4, pp. 662–667, 2009.

[159] O. Younis, M. Krunz, and S. Ramasubramanian, "Node clustering in wireless sensor networks: Recent developments and deployment challenges," *IEEE Netw.*, vol. 20, no. 3, pp. 20–25, May/Jun. 2006.

[160] G. Smaragdakis, I. Matta, and A. Bestavros, "SEP: A stable election protocol for clustered heterogeneous wireless sensor networks," Dept. Comput. Sci., Boston Univ., Boston, MA, USA, Tech. Rep. 2004-022, 2004.

[161] A.-S. Tonneau, N. Mitton, and J. Vandaele, "How to choose an experimentation platform for wireless sensor networks? A survey on static and mobile wireless sensor network experimentation facilities," *Ad Hoc Netw.*, vol. 30, pp. 115–127, Jul. 2015.

[162] J. Hill, M. Horton, R. Kling, and L. Krishnamurthy, "The platforms enabling wireless sensor networks," *Commun. ACM*, vol. 47, no. 6, pp. 41–46, Jun. 2004.

[163] B. Kirchen, M. H. Alizai, I. K. Wehrle, and I. S. Kowalewski, "TinyWifi: Enabling Linux platform support in TinyOS," Ph.D. dissertation, Chair Commun. Distributed Syst., RWTH Aachen Univ., Aachen, Germany, 2010.

[164] V. Handziski *et al.* (Feb. 22, 2007). *Hardware Abstraction Architecture*. [Online]. Available: http://www.tinyos.net/tinyos-2.x/doc/tep2, accessed 2012.

[165] J. L. Hill and D. E. Culler, "Mica: A wireless platform for deeply embedded networks," *IEEE Micro*, vol. 22, no. 6, pp. 12–24, Nov./Dec. 2002.

[166] S. Farshchi, P. H. Nuyujukian, A. Pesterev, I. Mody, and J. W. Judy, "A TinyOS-enabled MICA2-BasedWireless neural interface," *IEEE Trans. Biomed. Eng.*, vol. 53, no. 7, pp. 1416–1424, Jul. 2006.

[167] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Proc. 4th Int. Symp. Inf. Process. Sensor Netw. (IPSN)*, Apr. 2005, pp. 364–369.

[168] J. R. Agre, L. P. Clare, G. J. Pottie, and N. P. Romanov, "Development platform for self-organizing wireless sensor networks," *Proc. SPIE Int. Soc. Opt. Eng.*, vol. 3713, pp. 257–268, Jul. 1999.

[169] T. Paczesny, T. Tajmajer, J. Domaszewicz, and A. Pruszkowski, "ProxyMotes: Linux-based TinyOS platform for non-TinyOS sensors and actuators," in *Proc. IEEE 10th Int. Symp. Parallel Distrib. Process. Appl. (ISPA)*, Jul. 2012, pp. 255–261.

[170] M. Hempstead, M. Welsh, and D. Brooks, "TinyBench: The case for a standardized benchmark suite for TinyOS based wireless sensor network devices," in *Proc. 29th Annu. IEEE Int. Conf. Local Comput. Netw.*, Nov. 2004, pp. 585–586.

[171] M. Lin, Y. Wu, and I. Wassell, "Wireless sensor network: Water distribution monitoring system," in *Proc. IEEE Radio Wireless Symp.*, Jan. 2008, pp. 775–778.

[172] W.-Y. Chung and J.-H. Yoo, "Remote water quality monitoring in wide area," *Sens. Actuators B, Chem.*, vol. 217, pp. 51–57, Oct. 2015.

[173] A. Sikora, P. Digeser, M. Klemm, M. Tubolino, and R. Werner, "Model based development of a TinyOS-based wireless M-bus implementation," in *Proc. IEEE 1st Int. Symp. Wireless Syst. (IDAACS-SWS)*, Sep. 2012, pp. 91–94.

[174] J. Heidemann, W. Ye, J. Wills, A. Syed, and Y. Li, "Research challenges and applications for underwater sensor networking," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, vol. 1, Apr. 2006, pp. 228–235.

[175] J. Heidemann, M. Stojanovic, and M. Zorzi, "Underwater sensor networks: Applications, advances and challenges," *Philos. Trans. Roy. Soc. London A, Math. Phys. Sci.*, vol. 370, no. 1958, pp. 158–175, Jan. 2012.

[176] Y. Noh, D. Torres, and M. Gerla, "Software-defined underwater acoustic networking platform and its applications," *Ad Hoc Netw.*, vol. 34, pp. 252–264, Nov. 2015.

[177] N. Al-Nakhala, R. Riley, and T. Elfouly, "Distributed algorithms in wireless sensor networks: An approach for applying binary consensus in a real testbed," *Comput. Netw.*, vol. 79, pp. 30–38, Mar. 2015.

[178] M. Britton, V. Shum, L. Sacks, and H. Haddadi, "A biologically-inspired approach to designing wireless sensor networks," in *Proc. 2nd Eur. Workshop Wireless Sensor Netw.*, Jan./Feb. 2005, pp. 256–266.

[179] K. Sohraby, D. Minoli, and T. Znati, *Wireless Sensor Networks: Technology, Protocols, and Applications*. New York, NY, USA: Wiley, 2007.

[180] S. Farshchi, I. Mody, and J. W. Judy, "A TinyOS-based wireless neural interface," in *Proc. 26th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (IEMBS)*, vol. 2, Sep. 2004, pp. 4334–4337.

[181] S. Farshchi, P. H. Nuyujukian, A. Pesterev, I. Mody, and J. W. Judy, "A TinyOS-based wireless neural sensing, archiving, and hosting system," in *Proc. 2nd Int. IEEE Conf. EMBS Neural Eng.*, Mar. 2005, pp. 671–674.

[182] S. Farshchi, A. Pesterev, P. Nuyujukian, E. Guenterberg, I. Mody, and J. W. Judy, "Embedded neural recording with TinyOS-based wireless-enabled processor modules," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 18, no. 2, pp. 134–141, Apr. 2010.

[183] J. Li and G. Serpen, "nesC-TinyOS model for parallel and distributed computation of max independent set by Hopfield network on wireless sensor network," *Procedia Comput. Sci.*, vol. 6, pp. 396–401, Dec. 2011.

[184] W. Kiing-Ing, "A light-weighted, low-cost and wireless ECG monitor design based on TinyOS operating system," in *Proc. 6th Int. Special Topic Conf. Inf. Technol. Appl. Biomed. (ITAB)*, Nov. 2007, pp. 165–168.

[185] G. Tolle and D. Culler, "Design of an application-cooperative management system for wireless sensor networks," in *Proc. EWSN*, vol. 5, Jan./Feb. 2005, pp. 121–132.

[186] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *ACM SIGOPS Oper. Syst. Rev.*, vol. 35, no. 11, pp. 93–104, Nov. 2000.

[187] W. Stallings, G. K. Paul, and M. M. Manna, *Operating Systems: Internals and Design Principles*, vol. 3. Upper Saddle River, NJ, USA: Prentice-Hall, 1998.

[188] T. M. Cao, B. Bellata, and M. Oliver, "Design of a generic management system for wireless sensor networks," *Ad Hoc Netw.*, vol. 20, pp. 16–35, Sep. 2014.

[189] S. Hu, Y. Yu, and L. Xie, "Comparing power management strategies of Android and TinyOS," in *Proc. 3rd Pacific-Asia Conf. Circuits, Commun. Syst. (PACCS)*, Jul. 2011, pp. 1–4.

[190] N. Peterson *et al.*, "TinyOS-based quality of service management in wireless sensor networks," in *Proc. 42nd Hawaii Int. Conf. Syst. Sci. (HICSS)*, Jan. 2009, pp. 1–10.

[191] S. S. Bhunia, S. K. Das, S. Roy, and N. Mukherjee, "Mobility management in IP based wireless sensor network using TinyOS," in *Proc. 6th Int. Conf. Sens. Technol. (ICST)*, Dec. 2012, pp. 759–764.

[192] J. Guevara, E. Vargas, F. Brunetti, F. Barrero, and L. Aranda, "A framework for WSN using TinyOS and the IEEE1451 standard," in *Proc. IEEE Latin-Amer. Conf. Commun. (LATINCOM)*, Oct. 2011, pp. 1–5.

[193] T. Roosta, S. Shieh, and S. Sastry, "Taxonomy of security attacks in sensor networks and countermeasures," in *Proc. 1st IEEE Int. Conf. Syst. Integr. Rel. Improvements*, vol. 25, 2006, p. 94.

[194] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Ad Hoc Netw.*, vol. 1, nos. 2–3, pp. 293–315, Sep. 2003.

[195] D. R. Raymond and S. F. Midkiff, "Denial-of-service in wireless sensor networks: Attacks and defenses," *IEEE Pervasive Comput.*, vol. 7, no. 1, pp. 74–81, Jan./Mar. 2008.

[196] X. Zhu and Y. Chen, "Research of wireless injection attacks based on TinyOS," in *Proc. 3rd Int. Conf. Consum. Electron., Commun. Netw. (CECNet)*, Nov. 2013, pp. 525–528.

[197] J. K. Hart and K. Martinez, "Environmental sensor networks: A revolution in the earth system science?" *Earth-Sci. Rev.*, vol. 78, nos. 3–4, pp. 177–191, Oct. 2006.

[198] W.-S. Jang, W. M. Healy, and M. J. Skibniewski, "Wireless sensor networks as part of a Web-based building environmental monitoring system," *Autom. Construction*, vol. 17, no. 6, pp. 729–736, Aug. 2008.

[199] S. N. Simić and S. Sastry, "Distributed environmental monitoring using random sensor networks," in *Information Processing in Sensor Networks*. Berlin, Germany: Springer-Verlag, 2003, pp. 582–592.

[200] R. Gao, H. Zhou, and G. Su, "A wireless sensor network environment monitoring system based on TinyOS," in *Proc. Int. Conf. Electron. Optoelectron. (ICEOE)*, vol. 1, Jul. 2011, pp. V1-497–V1-501.

[201] M. Delamo, S. Felici-Castell, J. J. Pérez-Solano, and A. Foster, "Designing an open source maintenance-free environmental monitoring application for wireless sensor networks," *J. Syst. Softw.*, vol. 103, pp. 238–247, May 2015.

[202] D. Patnode, J. Dunne, A. Malinowski, and D. Schertz, "WISENET—TinyOS based wireless network of sensors," in *Proc. 29th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, vol. 3, Nov. 2003, pp. 2363–2368.

[203] C. Bin, J. Xinchao, Y. Shaomin, Y. Jianxu, Z. Xibin, and Z. Guowei, "Application research on temperature WSN nodes in switchgear assemblies based on TinyOS and ZigBee," in *Proc. 4th Int. Conf. Electr. Utility Deregulation Restruct. Power Technol. (DRPT)*, Jul. 2011, pp. 535–538.

[204] N. Wang, N. Zhang, and M. Wang, "Wireless sensors in agriculture and food industry—Recent development and future perspective," *Comput. Electron. Agricult.*, vol. 50, no. 1, pp. 1–14, Jan. 2006.

[205] T. Wark *et al.*, "Transforming agriculture through pervasive wireless sensor networks," *IEEE Pervasive Comput.*, vol. 6, no. 2, pp. 50–57, Apr./Jun. 2007.

[206] F. J. Pierce and T. V. Elliott, "Regional and on-farm wireless sensor networks for agricultural systems in Eastern Washington," *Comput. Electron. Agricult.*, vol. 61, no. 1, pp. 32–43, Apr. 2008.

[207] F. G. Montoya *et al.*, "A monitoring system for intensive agriculture based on mesh networks and the Android system," *Comput. Electron. Agricult.*, vol. 99, pp. 14–20, Nov. 2013.

[208] Y. Jihua and W. Wang, "Research and design of solar photovoltaic power generation monitoring system based on TinyOS," in *Proc. 9th Int. Conf. Comput. Sci. Edu. (ICCSE)*, Aug. 2014, pp. 1020–1023.

[209] P. Jishen, L. Wenshuai, and L. Qiuxiang, "Research and design of solar photovoltaic power generation wireless remote monitoring system," *Comput. Meas. Control*, vol. 20, no. 12, p. 30, 2012. DOI: 10.1109/ICCSE.2014.6926617

[210] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proc. 1st ACM Int. Workshop Wireless Sensor Netw. Appl.*, 2002, pp. 88–97.

[211] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," *Commun. ACM*, vol. 47, no. 6, pp. 34–40, Jun. 2004.

[212] J. A. Tarifa, J. M. Escano, M. A. Molina, and C. Bordons, "Local measurement of harmonics through a sensor network based on TinyOS," in *Proc. SICE Annu. Conf. (SICE)*, 2012, pp. 1029–1034.

[213] C. Schmitt, T. Kothmayr, B. Ertl, W. Hu, L. Braun, and G. Carle, "TinyIPFIX: An efficient application protocol for data exchange in cyber physical systems," *Comput. Commun.*, Jun. 2014. DOI: 0.1016/j.comcom.2014.05.012

[214] L. Chen, R. Yan, and Z. Ma, "TinyOS-based localization system design using accelerometer," in *Proc. 15th IEEE Int. Conf. Commun. Technol. (ICCT)*, Nov. 2013, pp. 511–518.

[215] E. Monmasson and M. N. Cirstea, "FPGA design methodology for industrial control systems—A review," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1824–1842, Aug. 2007.

[216] B. Stelte, "Toward development of high secure sensor network nodes using an FPGA-based architecture," in *Proc. 6th Int. Wireless Commun. Mobile Comput. Conf.*, 2010, pp. 539–543.

[217] D.-H. T. That, A.-V. Dinh-Duc, and K. Phan-Dinh, "Implementation of TinyOS on FPGA system," in *Proc. IEEE Region Conf. TENCON*, Nov. 2010, pp. 1456–1459.

[218] I. Park, H. Shin, J. Park, E. Jung, and D. Har, "Improvement of TINYOS implementation for small memory FPGA system," in *Proc. XIII-IBERCHIP Workshop IWS*, 2007. [Online]. Available: http://www.iberchip.net/iberchip2007/articulos/2/a/poster/1–pleastop-Ieryung%20Park-IMPROVEMENT_TINYOS_IMPLEMENTATION.pdf

[219] A. Rezgui and M. Eltoweissy, "Service-oriented sensor–actuator networks: Promises, challenges, and the road ahead," *Comput. Commun.*, vol. 30, no. 13, pp. 2627–2648, Sep. 2007.

[220] G. S. Ramachandran, S. Michiels, W. Joosen, D. Hughes, and B. Porter, "Analysis of sensor network operating system performance throughout the software life cycle," in *Proc. 12th IEEE Int. Symp. Netw. Comput. Appl. (NCA)*, Aug. 2013, pp. 211–218.

[221] P. Levis. (2006). *TinyOS 2.0 Overview*. [Online]. Available: http://www.tinyos.net/dist-2.0.0/tinyos-2.0.0/doc/html/overview.html

[222] M. H. Alizai, H. Wirtz, B. Kirchen, and K. Wehrle, "Portable wireless-networking protocol evaluation," *J. Netw. Comput. Appl.*, vol. 36, no. 4, pp. 1230–1242, Jul. 2013.

[223] M. H. Alizai, H. Wirtz, B. Kirchen, T. Vaegs, O. Gnawali, and K. Wehrle, "TinyWiFi: Making network protocol evaluation portable across multiple phy-link layers," in *Proc. 6th ACM Int. Workshop Wireless Netw. Testbeds, Experim. Eval. Characterization*, 2011, pp. 19–26.

[224] M. H. Alizai, B. Kirchen, J. B. Link, H. Wirtz, and K. Wehrle, "TinyOS meets wireless mesh networks," in *Proc. 8th ACM Conf. Embedded Netw. Sensor Syst.*, 2010, pp. 429–430.

[225] C. Duffy, U. Roedig, J. Herbert, and C. J. Sreenan, "A performance analysis of MANTIS and TinyOS," Dept. Comput. Sci., University College Cork, Ireland, Tech. Rep. CS-2006-27-11, 2006.

[226] J. Jeong and D. Culler, "Incremental network programming for wireless sensors," in *Proc. 1st Annu. IEEE Commun. Soc. Conf. Sensor Ad Hoc Commun. Netw. (SECON)*, Oct. 2004, pp. 25–33.

[227] C. Lynch and F. O'Reilly, "PIC-based TinyOS implementation," in *Proc. 2nd Eur. Workshop Sensor Netw.*, Istanbul, Turkey, Feb. 2005, pp. 378–385.

[228] A. Dunkels, J. Alonso, and T. Voigt, "Making TCP/IP viable for wireless sensor networks," in *Proc. Work-Prog. Session 1st Eur. Workshop Wireless Sensor Netw. (EWSN)*, Berlin, Germany, 2004, p. 16.

[229] A. Dunkels, "Full TCP/IP for 8-bit architectures," in *Proc. 1st ACM/Usenix Int. Conf. Mobile Syst., Appl. Services (MobiSys)*, San Francisco, CA, USA, 2003, pp. 85–98.

[230] A. Dunkels, T. Voigt, N. Bergman, and M. Jönsson, "The design and implementation of an IP-based sensor network for intrusion monitoring," in *Proc. Swedish Nat. Comput. Netw. Workshop*, Karlstad, Sweden, 2004, pp. 1–4.

[231] J. Sa Silva, R. Ruivo, T. Camilo, G. Pereira, and F. Boavida, "IP in wireless sensor networks issues and lessons learnt," in *Proc. 3rd Int. Conf. Commun. Syst. Softw. Middlew. Workshops (COMSWARE)*, Bengaluru, India, Jan. 2008, pp. 496–502.

[232] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, *Transmission of IPv6 Packets Over IEEE 802.15.4 Networks*. RFC-4944, Internet Engineering Task Force, Sep. 2007. [Online]. Available: http://tools.ietf.org/html/rfc4944

Muhammad Amjad received the B.S. degree in telecommunication system from Bahauddin Zakariya University Multan, Pakistan. He is currently pursuing the M.S. degree in computer science with the COMSATS Institute of Information Technology, Islamabad, Pakistan. He has also worked as a Research Associate under the HEC Startup Grant. His research interests include wireless sensor networks, wireless energy transfer, vehicular ad hoc network, and electric vehicles. He is also a Reviewer of the *IEEE Communications Magazine* (Elsevier), *Computers and Electrical Engineering Journal* (Elsevier), *Ad Hoc Networks*, the *Journal of Network and Computer Applications*, and *Wireless Networks Journal* (Springer).

Muhammad Sharif received the Ph.D. degree from the COMSATS Institute of Information Technology, Wah Cantonment, Pakistan. He is currently an Associate Professor with COMSATS. His area of specialization is image processing. He has been in the teaching field since 1995. He has more than 100 research publications in IF, SCI, and ISI journals and in national and international conferences. Until now, he has supervised 25 M.S. (CS) theses. He is currently supervising five Ph.D. (CS) students and CoSupervisor of five others. More than 200 undergraduate students had already been passed out after successful completion of their project work under his supervision. His research interests are image processing, computer networks and security, and algorithms design and analysis.

Muhammad Khalil Afzal received the B.S. and M.S. degrees in computer science from the COMSATS Institute of Information Technology, Wah Campus, Wah Cantonment, Pakistan, in 2004 and 2007, respectively, and the Ph.D. degree from the Department of Information and Communication Engineering, Yeungnam University, Korea, in 2014. He has served as a Lecturer at Bahauddin Zakariya University, Multan, Pakistan, from 2008 to 2009, and King Khalid University, Abha, Saudi Arabia, from 2009 to 2011. He is currently an Assistant Professor with the Department of Computer Science at COMSATS. His research interest includes wireless sensor networks, ad hoc networks, reliability in multicasting, and cooperative networks.

Sung Won Kim received the B.S. and M.S. degrees from the Department of Control and Instrumentation Engineering, Seoul National University, Korea, in 1990 and 1992, respectively, and the Ph.D. degree from the School of Electrical Engineering and Computer Sciences, Seoul National University, in 2002. From 1992 to 2001, he was a Researcher with the Research and Development Center, LG Electronics, Korea. From 2001 to 2003, he was a Researcher with the Research and Development Center, AL Tech, Korea. From 2003 to 2005, he was a Post-Doctoral Researcher with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, USA. In 2005, he joined the Department of Information and Communication Engineering, Yeungnam University, Gyeongsan, Korea, where he is currently a Professor. His research interests include resource management, wireless networks, mobile networks, performance evaluation, and embedded systems.