Mapping Wireless Sensor Network Applications Requirements to Existing Operating Systems

Tarek M.Salem[1], **Sherine M. Abd El-kader**[2]

[1] **Assistant Research at Electronics Research Institute, Computers and Systems Dept, Cairo, Egypt**

[2] **Associate Professor at Electronics Research Institute, Computers and Systems Dept, Cairo, Egypt**

Abstract
The design of operating system for Wireless Sensor Network (WSN) deviates from traditional operating system design due to significant and specific characteristics like constrained resources, high dynamics and inaccessible deployment. The purpose of this work is to classify existing operating systems according to the important Operating System (OS) features and to propose the suitable OSs for different categories of WSN applications. Architecture, execution model, scheduling, routing protocols, hardware support, and application support are the important OS features that are chosen to classify the existing WSN operating systems. This classification helps in understanding the contrasting differences between the existing operating systems and lays the foundation for designing an ideal operating system. To help the application developer in choosing the right OS, based on the application requirement, hardware type, also WSN applications have been classified. This classification gives insight in choosing the best suitable operating systems that fits for different categories of applications.

Keywords: *Wireless sensor network, Operating systems, Embedded operating system, Real-time operating system, Application requirements.*

1. Introduction

A wireless sensor node is a good example for a System on Chip (SoC) that has communication, computation, sensing and storage capabilities. These miniaturized nodes have stringent constraints in terms of available resources like processing power, battery power, program memory, available bandwidth. Basically, each node comprises of a micro-controller, power source, Radio Frequency (RF) transceiver, external memory, and sensors. These sensor nodes collectively form a Wireless Sensor Network, which is used in wide variety of applications now days [1] [2]. A WSN typically consists of hundreds or thousands of sensor nodes. These nodes have the capability to communicate with each other using multi-hop communication. Typical applications of these WSN include but not limited to monitoring, tracking, and controlling.
The basic functionality of an operating system is to hide the low-level details of the sensor node by providing a clear interface to the external world. Processor management, memory management, device management, scheduling policies, multi-threading, and multitasking are some of the low level services to be provided by an operating system. In addition to the services mentioned above, the operating system should also provide services like support for dynamic loading and unloading of modules, providing proper concurrency mechanisms, Application Programming Interface (API) to access underlying hardware, and enforce proper power management policies.
Though some of these are similar to the services provided by traditional operating systems, the realization of those services in WSN is a non-trivial problem, due to the constraints on the resource capabilities. Hence a suitable operating system is required for WSN to provide these functionalities to facilitate the user in writing applications easily with little knowledge of the low-level hardware details.
Due to the significance of an operating system for WSNs and the availability of a significant body of literature on it, study of search becomes necessary and useful at this stage. Also analyzing different applications, their characteristics and suggesting an ideal OS for those applications will help an application developer to choose an operating system and therefore select best sensor hardware for specific application. Although there are many papers that surveys the characteristics, applications, and communication protocols of WSNs [3], prior to this there are no studies that survey the operating systems of WSN for specific application.

The rest of this paper is organized as follows. In Section 2, describes related work for this work. Section 3 describes design requirements of an operating system for WSNS. A classification framework and a comprehensive survey of existing operating systems against this framework are presented in Sections 4, 5 respectively. Comparative analysis between different operating systems is tabulated in section 6. Conclusions and future work are introduced in Section 6, 7 respectively.

2. Related Work

Muhammad Omer, Thomas Kunz [4] have been investigated the most widely used operating systems for WSNs. Also understand the characteristics of popular OSs for WSN in particular and embedded devices in general without any studies about the relation between the OS and specific application.
Wei Dong, Xue Liu [5] have been examined the challenges of the OS design space. Then introduce constitutes a sensornet OS by describing its major components. Next, they provide an overview of the existing work, present a taxonomy of state-of-the-art sensornet OSes, and discuss various approaches to address the design challenges.
D. Manjunath [6] has been presented a well-rounded review of four popular operating systems proposed for WSNs: TinyOS, SOS, MANTIS, and Contiki. Inspired by the engineering approach, he has been identified the fundamental challenges involved in designing each component/feature of a typical sensorOS, and then described how these fundamental challenges have been approached by different sensor operating systems.
Hyunhak kim, Seongki [7] have been proposed an evolvable operating system for WSNs. Each component in this architecture is designed to perform its functionality concerned with power consumption, highly limited resources.
Margi, C.B, Escola de Artes [8] have been developed a comparison between two different operating systems which are Contiki and TinyOS that are running on the same hardware platform Crossbow TelosB. Using a set of tasks, which includes sensing, communication and security mechanisms, they have been evaluated their behavior in terms of energy consumption and execution time.
Ramon Serna, Ivan Shcherbakov, and others [9] have been presented an Operating System Abstraction Layer (OSAL) to reduce the portability efforts of software applications between platforms, independent of the running OS. They claimed that such design reduces dramatically the required efforts related to portability of application code, and effectively enables the re-utilization of software components in later deployments.
In this paper, mapping wireless sensor network applications requirements to existing operating systems will be introduced.

3. Design Issues and Challenges

WSN operates at two levels [10]. One is at the network level and the other is at node level. Network level interests are connectivity, routing, communication channel characteristics, and protocols. Node level interests are hardware, radio, CPU, sensors and limited energy. At a higher level OS for WSN can also be classified as node-level (local) and network-level (distributed). The important issues related to node-level are limited resource management, concurrency handling, power management and memory management where as issues related to both are inter-node communication, failure handling, heterogeneity and scalability.
This section discusses the important issues of both node and network-level to be considered while designing an operating system for WSN. These issues discuss the challenges and motivate the design requirements of an operating system needed for WSN.

3.1 Restricted Resource

A typical sensor node shown in figure 1 is constrained by the resources available to it. It is constrained by limited battery power, processing capability, memory and bandwidth.

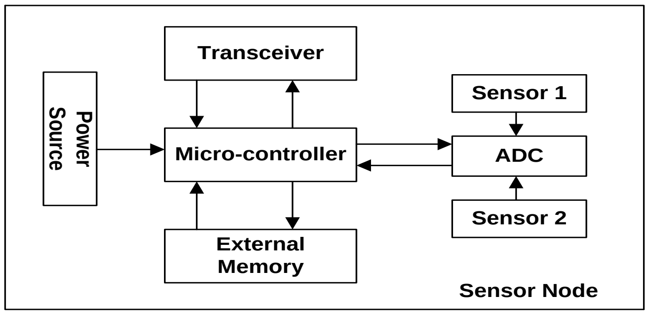

Fig. 1 Sensor Node Architecture

Figure 2 depicts, where operating system stands in the software layers of the WSN. Middleware and application layers are distributed across the nodes. Core kernel of the operating system sits at each individual node. On top of it, middleware and applications run as interacting modules across nodes.

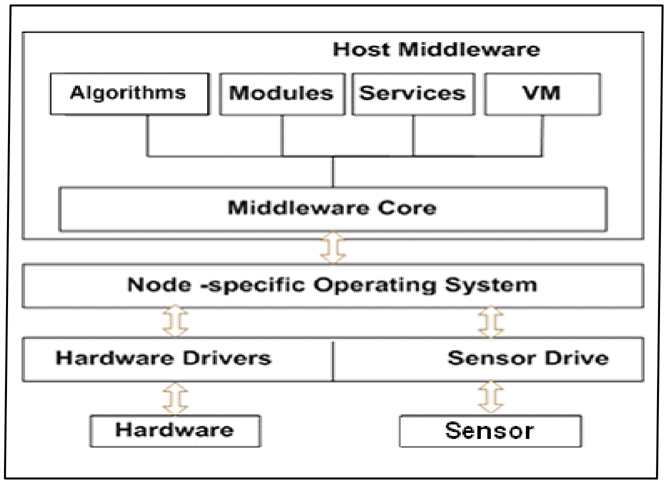

Fig. 2 Software Layers

3.1.1 Battery power

Power consumption is crucial to the life span of WSN based applications. Most of the applications in WSN are long lived ranging from days to years. So a typical node with a limited power supply has to live mostly for months to years. Unlike conventional systems where the power is not at all a constraint factor in building the system, operating systems for sensor nodes have to consider power as one of the available resources like processor and memory.

3.1.2 Processing Power

Sensor nodes will have a processing power in the order of a few MIPS. Computation intensive operations should be properly scheduled; otherwise high priority tasks get delayed/starved. Computation models like event driven will follow run-to-completion model. This takes more processor time if the task is running for long time and preventing other jobs to wait for longer time irrespective of their priorities. Hence operating system should properly schedule the processor according to the priority of jobs.

3.1.3 Memory

The current generation of micro-controllers family such as Mica [11], its successors and some microcontrollers (e.g. nymph, EYES etc) specific to various research projects have nearly 128kbytes of program memory. One of the main constraints for the developer is this available program memory and operating system developed for WSN should fit within this memory. The system software such as operating system, virtual machine, middleware, and application algorithms have to fit into this memory. Optimal usage of this memory should start from lower level (i.e. Operating System).

3.1.4 Bandwidth

A typical sensor node uses RF channel to communicate with other sensor nodes in the network. ZigBee [12] is the emerging standard to define the communication protocol stack based on the existing physical and data-link layers of IEEE 802.15.4 Personal Area Network (PAN) standard. Data rate supported by PANs is 256kbps. Whereas Bluetooth standard supports data rate up to 3Mbps. CC1000 is another standard that has been widely used in sensor networks. Its data rate is around 39kbps.

3.2 Portability

The hardware platforms in WSN are evolving day-by-day. Portability is an important issue to be considered as everyone is working on their customized hardware platforms. Portability is one of the main concerns for the developer to make the software work on different hardware platforms. The operating system should be written in such a way that it is easily portable to different hardware platforms with minimal changes.

3.3 Customizability

Applications in WSN are spread over different disciplines. Specific applications of WSN include but not limited to monitoring environment, surveillance, target tracking etc. Survey of some of these applications can be found in [12] [13] [14]. Most of the software platforms developed for WSN are application specific. Different applications demand different requirements from operating system. These requirements may be reconfigurability, real-time guarantees. The design of OS should be in such a way that it should be easily customizable and extensible to various applications.

3.4 Multicasting

At a given point of time, nodes in the WSN could be doing more than one task. For example, consider a typical application where in the sensed data from the environment is collected, aggregated based on some filtering conditions, encrypted/decrypted and passed it towards the sink node through other nodes. In this application the sensor node has to do the following tasks at a given point of time:

- Sense the data.
- Collect data from other neighborhood sensor nodes.
- Aggregate the data based on the certain conditions provided.
- Route the data to the sink node.

3.5 Network Dynamics

Mobility, failure of communication channels/nodes constitutes the dynamics in WSN. Topologies are more prone to changes due to these dynamics which may result in network partitions. Link failures and the interferences in the RF communication channel deviates the behavior of the WSN from its normal operation. Operating system should adapt the application according to the context of different dynamics of the environment. This helps in providing transparency from network dynamics to the application.

3.6 Distributed Nature

There is a clear distinction between the services that should be supported by middleware and OS in traditional systems. This is masked in WSN due to cross layer interaction support which is a prominent feature for these kinds of systems. Sensor nodes in WSN are loosely coupled and sometimes deployed across a large geographical area. The scale of the network sometimes is in the order of thousands of sensor nodes. Each individual node has its own processing power, system software to run and the co-operation among the nodes happen through exchange of messages.

3.6.1 Heterogeneity

Heterogeneity in the network arrives due to varying level of node capabilities. This causes different nodes to be present in the network with different capabilities. These capabilities can be in terms of memory, sensing modality or residual energy, software components residing at the node. Many of the practical sensor networks are heterogeneous [14] in their sensing capability.

3.6.2 Scalability

Scalability here refers to the size of the network [15]. As the system is composed of large number of nodes, the system algorithms should work with acceptable performance degradation with increase in the number of nodes. In WSN there are subtle differences related to some issues in designing middleware and operating system. Some of the above issues seems to look like middleware issues but virtual machine approaches [12] and distributed operating systems [16] in WSN did concerned about them in designing.

4. Designed Characteristics

The following are the important design characteristics to be considered while designing an operating system for WSN.

4.1 Architecture

The organization of an OS constitutes its structure. The architecture of an OS has an influence on the size of the OS kernel as well as on the way it provides services to the application programs. Some of the well known OS architectures are the monolithic architecture, the micro-kernel architecture, the virtual machine architecture and the layered architecture.

A monolithic architecture [9] in fact does not have any structure. Services provided by an OS are implemented separately and each service provides an interface for other services. Such an architecture allows bundling of all the required service together into a single system image, thus results in a smaller OS memory footprint. An advantage of the monolithic architecture is that the module interaction costs are low.

An alternate choice is a microkernel architecture [9] in which minimum functionality is provided inside the kernel. Thus, the kernel size is significantly reduced. Most of the OS functionality is provided via user-level servers like a file server, a memory server, a time server, etc. If one server fails, the whole system does not crash. The microkernel architecture provides better reliability, ease of extension and customization.

A virtual machine [9] is another architectural choice. The main idea is to export virtual machines to user programs, which resemble hardware.

4.2 Efficient Execution Model

The execution model provides the abstraction of computational unit and defines services like synchronization, communication, and scheduling. These abstractions are used by the programmer for developing applications. Communication service defines the way the computational units communicate. They communicate to exchange data, delegation of functionalities and signaling. While communicating there can be data that is shared. Accessing shared data requires proper synchronization mechanisms to avoid race conditions.

4.3. Communication Protocol Support

In the OS context, communication refers to inter-process communication within the system as well as with other nodes in the network. WSNs operate in a distributed environment, where senor nodes communicate with other nodes in the network. All WSN OSs provide an Application Programming Interface (API) that enables application program to communicate. It is possible that a WSN is composed of heterogeneous sensor nodes, therefore the communication protocol provided by the OS must also consider heterogeneity.

4.4. Programming Model

The programming model supported by an OS has a significant impact on the application development. There are two popular programming models provided by typical WSN OSs, namely: event driven programming and multithreaded programming. Multithreading is the application development model most familiar to programmer, but in its true sense rather resource intensive, therefore not considered well suited for resource constraint devices such as sensor nodes. Event driven programming is considered more useful for computing devices equipped with scarce resource but not considered convenient for traditional application developers. Therefore researchers have focused their attention on developing a light-weight multithreading programming model for WSN Oss.

4.5. Scheduling

The Central Processing Unit (CPU) scheduling determines the order in which tasks are executed on a CPU. In traditional computer systems, the goal of a scheduler is to minimize latency, to maximize throughput and resource utilization, and to ensure fairness. The selection of an appropriate scheduling algorithm for WSNs typically depends on the nature of the application. For applications having real-time requirements, real-time scheduling algorithm must be used. For other applications, non-real-time scheduling algorithms are sufficient. WSNs are being used in both real-time [17] [18] and non-real-time [19] environments; therefore a WSN OS must provide scheduling algorithms that can accommodate the application requirements.

4.6 Resource Management
One of the fundamental tasks of an operating system is to manage the system resources efficiently. Resources available in a typical sensor node are processor, program memory, battery, and sensors etc. Efficient use of processor involves using a scheduler with optimal scheduling policy. Usage of memory involves memory protection, dynamic memory allocation, etc. Battery should be treated as a special resource. Sleep modes help in power management of battery. Managing sensors include controlling sensing rate.

5. Classification framework for WSN Operating Systems

Architecture, execution model, scheduling, routing protocols, hardware support, and application support have been chosen as the important design features that forms basis for our classification framework.
A different type of each feature is depicted in the figure 3. Based on each feature certain operating system, suitable hardware, and suitable routing protocol could be suggested. Miscellaneous features shown in the figure are explained for each operating system. These features are simulation support and programming language. Below is an overview of the design features.

5.1. Classification According to Architecture
Architecture of the kernel influences the way it provides services as described in section 4.1.
Operating systems which fall under monolithic is TinyOS [16]. Also suitable hardware for monolithic is Mica2. Suitable routing protocol for monolithic is Sensor Protocol for Information Negotiation (SPIN). While operating systems which fall under modular is LiteOS [20].
Also suitable hardware for modular is Micaz. Suitable routing protocol for modular is MAC protocol. Operating systems which fall under VM is ContikiVM [21]. Also suitable hardware for VM is MSP430. Suitable routing protocol for it is multi-hop mesh routing as shown in figure 3. Architecture for different OS described in figure 4.

5.2 Classification According to Execution Model
Operating systems which fall under event-based is SOS [10]. Also suitable hardware for it is Tmote-sky. Suitable routing protocol for event-based execution model is directed diffusion. While operating systems which fall under thread-based is MantisOS [22]. Also suitable hardware for it is Mica2. Suitable routing protocol for thread-based is Sequential Assignment Routing (SAR). Operating systems which fall under Hybrid is Contiki [21]. Also suitable hardware for Hybrid is Avr MCU. Suitable routing protocol for it is Gradient-Based Routing (GBR) as shown in figure 3. Execution model of different Oss described in figure5.

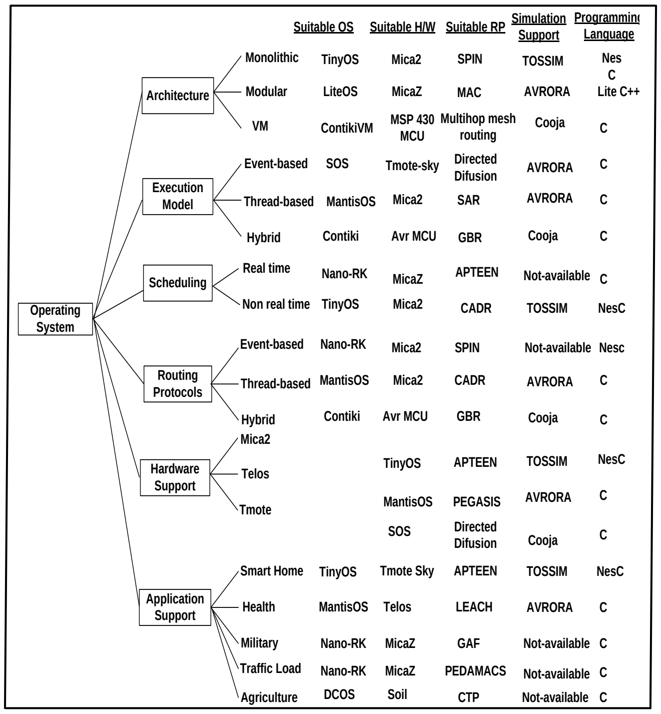
Suitable OS
Suitable H/W
Suitable RP
Simulation Support
Programming Language
Operating System
Architecture
Monolithic TinyOS Mica2 SPIN TOSSIM Nes C
Modular LiteOS MicaZ MAC AVRORA Lite C++
VM ContikiVM MSP 430 MCU Multihop mesh routing Cooja C
Execution Model
Event-based SOS Tmote-sky Directed Difusion AVRORA C
Thread-based MantisOS Mica2 SAR AVRORA C
Hybrid Contiki Avr MCU GBR Cooja C
Scheduling
Real time Nano-RK MicaZ APTEEN Not-available C
Non real time TinyOS Mica2 CADR TOSSIM NesC
Routing Protocols
Event-based Nano-RK Mica2 SPIN Not-available Nesc
Thread-based MantisOS Mica2 CADR AVRORA C
Hybrid Contiki Avr MCU GBR Cooja C
Hardware Support
Mica2
Telos
Tmote
TinyOS APTEEN TOSSIM NesC
MantisOS PEGASIS AVRORA C
SOS Directed Difusion Cooja C
Application Support
Smart Home TinyOS Tmote Sky APTEEN TOSSIM NesC
Health MantisOS Telos LEACH AVRORA C
Military Nano-RK MicaZ GAF Not-available C
Traffic Load Nano-RK MicaZ PEDAMACS Not-available C
Agriculture DCOS Soil CTP Not-available C

Fig. 3 Classification Framework for WSN operating system

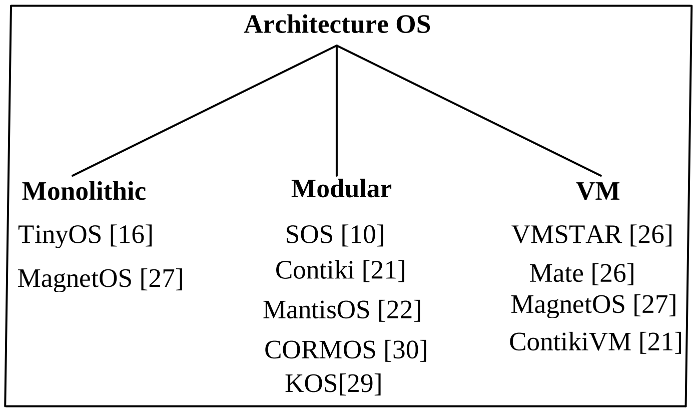

Fig. 4 Architecture of Different OSs

5.3. Classification According to Scheduling
Operating systems which fall under real time scheduling is Nano-RK [17]. Also suitable hardware for it is MicaZ.
Suitable routing protocol for real time scheduling is Threshold Sensitive Energy Efficient Sensor Network Protocol (APTEEN). While operating systems which fall under non-real time scheduling is TinyOS[16]. Also suitable hardware for it is Mica2. Suitable routing protocol for non-real time scheduling is Constrained Anisotropic

diffusion routing (CADR) as shown in figure 3. Read time and non-real time OS classification described in figure 6.

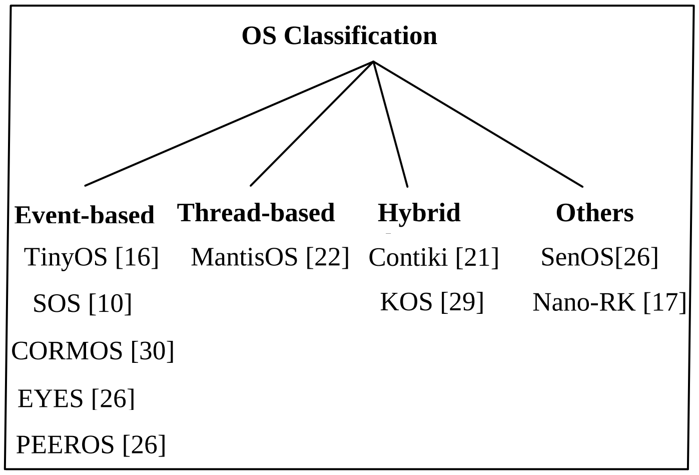

Fig. 5 Execution Model of Different OSs

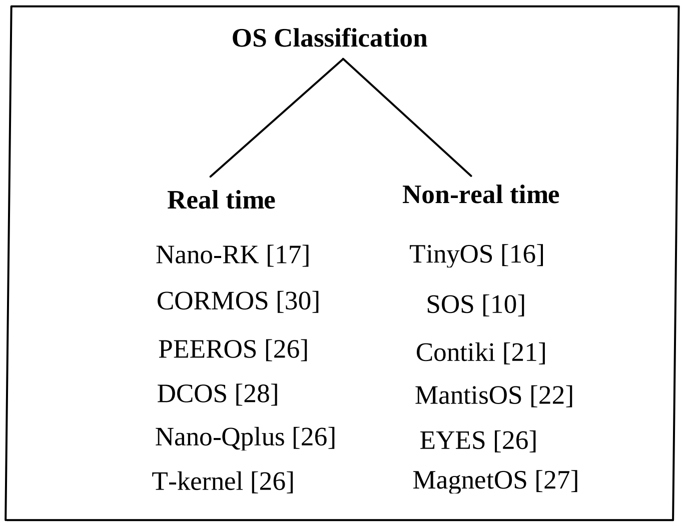

Fig. 7 Real-time and Non-real-time OS Classification

5.4. Classification According to Routing Protocols

Routing protocols can be classified into event based routing protocols [37], thread based routing protocols [38], and hybrid routing protocols [39]. The Example for event based is directed diffusion where the network will be active only when an event occur, otherwise the network is idle. The example for thread based is CADR. Example for hybrid routing protocol is GBR. Figure 3 presents the suitable hardware and operating system for each type of routing protocol.

Operating systems which fall under event-based is Nano-RK [17]. Also suitable hardware for it is Mica2. Suitable routing protocol for event-based is Sensor Protocol for Information Negotiation (SPIN). While operating systems which fall under thread-based is MantisOS [22]. Also suitable hardware for it is Mica2. Suitable routing protocol for thread-based is CADR. Operating systems which fall under Hybrid is Contiki [21]. Also suitable hardware for Hybrid is Avr MCU. Suitable routing protocol for it is GBR as shown in figure 3.

5.5. Classification According to Hardware Support.

There are many types of hardware WSN. Each type of hardware has suitable operating system. So the first thing to design any application is choose suitable sensor for application target. For example in real application the type of hardware differs from non-real applications.

Operating systems which fall under Mica2 hardware is TinyOS [10]. Also suitable routing protocol for it is APTEEN. While operating systems which fall under Telos hardware is MantisOS [22]. Suitable routing protocol for it is Power efficient Gathering Sensor Information System (PEGASIS).Operating systems which fall under Tmote hardware is SOS [10]. Also suitable routing protocol for it is directed diffusion as shown in figure 3. The lists of platforms supported as of now are shown in figure 7.

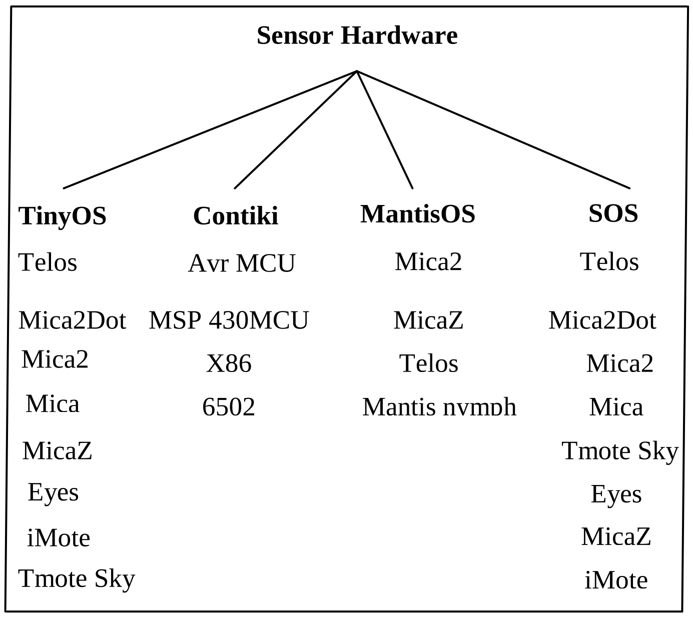

Fig. 7 Hardware Platforms Supported by different OS

5.6.Classification According to Application Support

Classification of applications depends on the nature of the purpose they are used for. In environment monitoring, the application can be broadly classified into indoor and outdoor monitoring [14]. While in Health application Involves monitoring patients and alerting doctors. Here sensors measure the recent actions of the patients and the Remind the doctors about the behavior of the patient.

Category	Suitable H/W	Suitable OS	Suitable RP	Topology	Programming Language
Environment Monitoring	Tmote	TinyOS	DD	Multi-hop	nesC
Health	Telos	MantisOS	LEACH	Cluster-head	C
Military	MicaZ	Nano-RK DCOS	GAF	Multi-hop	C
Agriculture	Soil sensor	Nano-RK	CTP	Tree-topology	C
Smart Home	Tmote Sky	TinyOS	APTEEN	Three tier	nesC
Traffic Load	MicaZ	T-kernel	PEDAMACS	Multi-hop	C
Habitat Monitoring	Mica2	SOS	SPAN	Cluster-head	C
Production/ Commercial	Telos	TinyOS	SAR	Three tier	nesC

Table 1: Application Characteristics and Suitable OSs

Another application might be tracking patients, doctors, and drug usage in the hospitals. In military application, Sensor nodes are well suited to the need of military applications. Interesting of them are information collection, enemy tracking, battlefield surveillance, target classification, perimeter security, border patrol. Other applications are smart home [23], Agriculture [24], and Traffic Load [32].
Characteristics of applications are evaluated against the categories of applications, suitable hardware, suitable operating system, suitable routing protocol, type of topology and programming language as shown in Table 1.
Suitable hardware which falls under environment application is Tmote. Also suitable OS for it is TinyOS. Suitable routing protocol for it is Directed Diffusion (DD). Type of topology for this application is multi-hop. Suitable hardware which falls under health application is Telos. Also suitable OS for it is MantisOS. Suitable routing protocol for it is LEACH. Type of topology for this application is cluster-head. Suitable hardware which falls under military application is MicaZ. Also suitable OS for it is DCOS [26]. Suitable routing protocol for it is Geographic Adaptive Fidelity (GAF). Type of topology for this application is multi-hop as shown in Table 1.
Suitable hardware which falls under agriculture application is Soil sensor. Also suitable OS for it is Nano-RK. Suitable routing protocol for it is Collection Tree Protocol (CTP). Type of topology for this application is tree-topology.
Suitable hardware which falls under smart home application is Tmote. Also suitable OS for it is TinyOS. Suitable routing protocol for it is Adaptive Threshold Sensitive Energy Efficient Sensor Network Protocol (APTEEN). Type of topology for this application is three-tier.
Suitable hardware which falls under traffic load application is MicaZ. Also suitable OS for it is T-kernel [26]. Suitable routing protocol for it is Power Efficient and Delay Aware Medium Access Protocol (PEDAMACS). Type of topology for this application is multi-hop.
Suitable hardware which falls under habitat application is Mica2. Also suitable OS for it is SOS. Suitable routing protocol for it is SPAN. Type of topology for this application is cluster-head.
Suitable hardware which falls under production application is Telos. Also suitable OS for it is TinyOS. Suitable routing protocol for it is SAR. Type of topology for this application is three-tier as shown in Table 1.

6. Comparative analysis of Operating Systems

Summary of the existing operating systems such as TinyOS, SOS, Contiki, Mantis, and others presented in Table 2. The supported Features by them such as execution model, real time guarantee, dynamic programming, priority-based scheduling, and application supported are shown in Table 2.
Execution model which fall under TinyOS is component-based. Which no have real time guarantee, priority scheduling, and dynamic programming. Suitable application for it is smart home.
In another type, Execution model which fall under SOS is module-based. Which have priority scheduling, and dynamic programming. Suitable application for it is habitat monitoring. Also, Execution model which fall under Contiki is Hybrid. Which no have real time guarantee, and priority-based scheduling. Suitable application for it is temperature and humidity.

OS	Execution Model	Real Time Guarantee	Dynamic Reprogramming	Priority-based Scheduling	Application
TinyOS [16]	Component Based	No	No	No	Smart Home [23]
SOS[10]	Module based	No	Yes	Yes	Habitat monitoring [33]
Contiki [21]	Hybrid	No	Yes	No	Temperature and humidity [35]
Mantis [22]	Thread based	No	Yes	No	Health [31]
EYES [26]	Event Based	No	No	Yes	Light [25]
Nano-RK [17]	Task Based	Yes	Yes	Yes	Agriculture [24]
DCOS [28]	Data centric	Yes	Yes	No	Military [34]
MagnetOS [27]	VM based	Yes	No	No	Biomedical [36]
SenOS [26]	State based	No	Yes	No	Time-critical application [34]
T-kernel	Task Based	Yes	Yes	Yes	Traffic Control [32]

Table 2: Summary of Operating Systems

In MantisOS, Execution model which fall under it is thread-based. Which no have real time guarantee, and dynamic programming.
Suitable application for it is health. In EYES, Execution model which fall under it is event-based. Which no have real time guarantee, and priority-based scheduling. Suitable application for it is light. Execution model which fall under Nano-RK is task-based. Which have real time guarantee, and dynamic programming. Suitable application for it is agriculture.
In another type, Execution model which fall under DCOS is data-centric. Which have priority scheduling, and dynamic programming. Suitable application for it is military. In MagnetOS, Execution model which fall under it is VM-based. Which no have dynamic programming, and priority-based. Suitable application for it is biomedical. Finally, Execution model which fall under SenOS it is state-based. Which no have real time guarantee, and priority-state. Suitable application for it is time-critical.

7. Conclusions

This paper studies in depth the different operating systems related to WSN, then it maps between the WSN applications and the suitable OS according to the application's requirements. The contribution of this work is two-fold, the first contribution of this work is deriving and forming a classification framework for the existing OSs According to a lot of noteworthy important OS features, such as, architecture, Execution Model, Scheduling, and Routing Protocols, this classification helped in understanding the contrasting differences among existing operating systems, and lays a foundation to design an ideal WSN OS. The second contribution is offering a guide for selecting the appropriate OS to the desired WSN application where also the existing WSN applications have been classified and the applications categories have been mapped to the OS features. For example operating system which falls under agriculture application is Nano-RK, Whereas, the suitable OS for the time critical application is SOS. This helps the application developer in choosing the appropriate OS, based on the application requirements.

8. Future Work

Because of the potential applications of WSN, and growing interest on WSNs enforced to deploy multiple applications simultaneously on the same network. It gives an advantage of reducing infrastructure deployment cost. But, it is an interesting research challenge. Till now, there are not any OSs that facilitate running multiple applications. Though other desirable characteristics like low memory foot print and portability are mandatory, the above are the important objectives to be considered while designing new OS for an embedded WSN systems.

References

[1] Zhang, Junqi, Varadharajan, and others, "Wireless sensor network key management survey and taxonomy", Journal of network and computer applications, Vol. 33, Issue 2, pp.63-75, 2010.
[2] Basma M. Mohammad El-Basioni, Sherine M. Abd El-kader, Hussein S. Eissa, and Mohammed M. Zahra, "Clustering in Wireless Sensor Network: A Study on Three Well-known Clustering Protocols" Accepted as a book chapter in Developments in Wireless Network Prototyping, Design and Deployment: Future Generations, M. A. Matin (Ed.), IGI Global, 2013.
[3] Basma M. Mohammad, Sherine M. Abd El-kader, and Hussein S. Eissa, "Designing a local path repair algorithm for directed diffusion protocol", Egyptian Informatics Journal, Vol. 13, No. 3, pp. 155–169, November 2012.
[4] Muhammad Omer Farooq, and Thomas Kunz, "Operating Systems for Wireless Sensor Networks: A Survey ", MDPI Open Access Information and Policy, Vol. 11, No. 6, 2011.
[5] Wei Dong, and Xue Liu, "Providing OS Support for Wireless Sensor Networks: Challenges and Approaches", IEEE Communication Surveys and Tutorials, Vol. 12, No. 4, November 2010.
[6] D. Manjunath, "A Review of Current Operating Systems for Wireless Sensor Networks", In Proceedings of the 3th European Conference on Wireless Sensor Networks, Vol. 2, No. 1, pp. 293-308, January 2006
[7] Thu-Thuy, Do, Daeyoung Kim, Tomás Sánchez López, and Hyunhak Kim, "An Evolvable Operating System for Wireless Sensor Networks", International Journal of Software Engineering and Knowledge Engineering, Vol. 15, No. 1 , 2009.
[8] Margi, Escola de Artes, Cienc. E Humanidades, and others, "Impact of Operating Systems on Wireless Sensor Networks (Security) Applications and Testbeds", In Proceedings of the 19th International Conference on Computer Communications and Networks, Vol. 6, No. 2, , 2010.
[9] Ramon Serna, van Shcherbakov, and Gerhard Fohler, "An Operating System Abstraction Layer for Portable Applications in Wireless Sensor Networks", SAC '10 Proceedings of the 2010 ACM Symposium on Applied Computing, Vol. 10, No. 1, 2010.
[10] Shripad V Deshpande, and P. R. Devale, "Recent Trends in Using Wireless Sensor Networks in Industrial Environment", International Journal of Computer Networking, Vol. 3, No. 3, 2013.
[11] Hyoseung, and Hojung Cha, "Multithreading Optimization Techniques for Sensor Network Operating Systems". In Proceedings of the 4th European Conference on Wireless Sensor Networks, Vol. 4, No. 2, pp. 293-308, January 2007.
[12] M. Hussnain, M. Sharjeel, S. R. Chaudhry, S. A. Hussain, and others, "Investigating Multi-Topological ZigBee Based Wireless Sensor Network in Precision Agriculture", Journal of Basic and Applied Scientific Research, Lahore, Pakistan, Vol.3, No. 2, 2013.
[13] Basil Hamed, "Design and Implementation of Smart House Control Using Lab VIEW", International Journal of Soft Computing and Engineering (IJSCE), Vol. 1, No. 6, January 2012.
[14] Mohammed A. Hussein, "A Novel Approach for Indoor Outdoor Air Pollution Monitoring"; IJCSI International Journal of Computer Science Issues, Vol. 9, No. 4, University of Sulaimani, July 2012.
[15] Chris Karlof, Naveen Sastry , and David, "TinySec: A Link Layer Security for Wireless Sensor Networks", In Proceedings of the 2th ACM SenSys, Baltimore, MD, Vol.3, No. 5, November 2004.
[16] Cooprider, Archer, Eide, and others, "Efficient Memory Safety for Tinyos". In Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (SenSys'07), Vol.4, No. 3, pp. 205-218, New York, NY, USA, November 2007.
[17] Eswaran, Rowe, Rajkumar, and others, "Nano-RK: An Energy-Aware Resource-Centric RTOS for Sensor Networks". In Proceedings of the 26th IEEE Real-Time Systems Symposium, Miami, USA, Vol. 2, No. 1, December 2005.
[18] Rowe, Mangharam, Rajkumar, and R. FireFly, "A Time Synchronized Real-Time Sensor Networking Platform". In Proceedings of the 28th IEEE Real-Time Systems Symposium, Tucson, USA, Vol. 3, No. 7, 2007.
[19] Rowe, Lakshmanan, and Yhu, Rajkumar, "Rate-Harmonized Scheduling for Saving Energy". In Proceedings of the 29th IEEE Real-Time Systems Symposium, Barcelona, Spain, Vol.2, No. 2, 2008.
[20] Cao, Abdelzaher, Stankovic, and others "The LiteOS Operating System: Towards UNIX like Abstraction for Wireless Sensor Networks". In Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN), USA, Vol. 5, No. 2, April 2008.
[21] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt, "Contiki – A Lightweight and Flexible Operating System for Tiny Networked Sensors", In Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, Vol.2 No. 1, 2004.
[22] Bhatti, Carlson, Dai, Deng, and others, "Mantis OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms", Mobile Network. Appicationl, Vol. 10, Issue 2, pp. 563-579, 2005.
[23] Basma M. Mohammad , Sherine M. Abd El-kader, and Mahmoud Abdelmonim Fakhreldin, "Smart Home Design using Wireless Sensor Network and Biometric Technologies", International Journal of Application or Innovation in Engineering and Management (IJAIEM), Volume 2, Issue 3, pp. 413–429, March 2013.

[24] Sherine M. Abd El-kader and Basma M. Mohammad El-Basioni, "Precision Farming Solution in Egypt Using the Wireless Sensor Network Technology", Egyptian Informatics Journal, Volume 14, Issue 3, July 2013.
[25] A. Mason1, L. E. Cordova Lopez, and A. Shaw, "Road traffic management using GIS and WSN", Built Environment and Sustainable Technologies, Liverpool John Moores University, Liverpool, UK, Vol.4, Issue 2, 2011.
[26] A. K. Dwivedi, M. K. Tiwari, and O. P. Vyas, "Operating Systems for Tiny Networked Sensors: A Survey", International Journal of Recent Trends in Engineering, Vol. 1, Issue 2, May 2009.
[27] J.Radhika, and S.Malarvizhi, "Middleware approaches for Wireless Sensor Networks: An overview", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 6, No 3, November 2012.
[28] T. Hofmeijer, S. Dulman, P. G. Jansen, and P. J. M. Havinga. "Dcos, a real-time light-weight data centric operating system". Int. Conf. on (ACST), St. Thomas, Virgin Islands, USA, pp. 259-264, Calgary, Canada, November 2004.
[29] Yan Liang; Jiannong Cao; Zhang, D.; Rui Wang; Quan Pan "A Biologically Inspired Sensor Wakeup Control Method for Wireless Sensor Networks", Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, On pp. 525 – 538 Volume: 40, Issue: 5, Sept. 2010.
[30] J. Yannakopoulos and A. Bilas. Cormos: A communication-oriented runtime system for sensor networks. In The Second European Workshop on Wireless Sensor Networks (EWSN 2005), February 2005.
[31] J. Stankovic, Q. Cao, T. Doan, L. Fang, Z. He, R. Kiran, S. Lin, S. Son, R. Stoleru, and A. Wood. "Wireless sensor networks for in-home healthcare: Potential and challenges". In High Confidence Medical Device Software and Systems (HCMDSS) Workshop, Philadelphia, PA, June 2005.
[32] Maha Mohamed Nabeel, Mahmoud Fakher el Deen, Sherine Abd El-Kader, "Intelligent Vehicle Recognition based on Wireless Sensor Network," International Journal of Computer Science Issues (IJCSI), Volume 10, Issue 4, July 2013
[33] Tomasz Naumowicz, Robin Freeman, and others, "Wireless Sensor Network for habitat monitoring on Skomer Island", In proceeding of: The 35th Annual IEEE Conference on Local Computer Networks, LCN 2010, 10-14 October 2010, Denver, Colorado, USA.
[34] Tian He, John A. Stankovic, and others, "Energy-efficient surveillance system using wireless sensor networks", In Proceedings of the 2nd international conference on Mobile systems, applications, and services, pp.270-283, ACM New York, NY, USA, 2004.
[35] Kshitij Shinghal 1, Arti Noor, and others, "Intelligent Humidity Sensor for Wireless Sensor Network Agriculture Application", International Journal of Wireless and Mobile Networks (IJWMN) Vol. 3, No. 1, February 2011.
[36] Loren Schwiebert, Sandeep K.S, and others, "Research challenges in wireless networks of biomedical sensors", In Proceedings of the 7th annual international conference on Mobile computing and networking, pp. 151-165, ACM New York, NY, USA, 2001.
[37] Anar A. Hady *, Sherine M. Abd El-kader, Hussein S. Eissa, "Intelligent Sleeping Mechanism for wireless sensor networks", Egyptian Informatics Journal, ISSN 1110-86652013, April 2013.
[38] T.salem, Salah.Abd-Elmagied, and others "Improving the Routing Mechanism of Clusters Sensor Networks", International Journal of Intelligent Computing and Information Science, Vol.10, Issue 3, July 2012.
[39] Venugopalan Ramasubramanian , Zygmunt J. Haa, "SHARP: A Hybrid Adaptive Routing Protocol for Mobile Ad Hoc Networks", In Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing, pp. 303-314, ACM New York, USA, 2003.

AUTHOR

Tarek M. Salem is an assistant researcher in Computers and Systems Department at the Electronics Research Institute (ERI) in Egypt. In May 2013, he completed her B.S. in Computers and Systems department, Faculty of Engineering, Al-Azhar University. During the 2005-2012 year, he joined the Arabic Organization for Industrialization. In 2013 year, he occupied the position of research assistant at Electronics Research Institute. He is a Lecturer, at Aljazeera Institute for information technology from 2011 till now. Now, he is studying for Ph.D. degree.

S. Abd El-kader has her MSc, & PhD degrees from the Electronics & Communications Dept. & Computers Dept., Faculty of Engineering, Cairo University, at 1998, & 2003. Dr. Abd El-kader is an Associate Prof., Computers & Systems Dept., at the Electronics Research Institute (ERI). She is currently supervising 3 PhD students, and 10 MSc students. Dr. Abd El-kader has published more than 25 papers, 4 book chapters in computer networking area. She is working in many computer networking hot topics such as; Wi-MAX, Wi-Fi, IP Mobility, QoS, Wireless sensors Networks, Ad-Hoc Networking, realtime traffics, Bluetooth, and IPv6. She is an Associate Prof., at Faculty of Engineering, Akhbar El Yom Academy from 2007 till 2009. Also she is a technical reviewer for many international Journals. She is heading the Internet and Networking unit at ERI from 2003 till now. She is also heading the Information and Decision making support Center at ERI from 2009 till now. She is supervising many automation and web projects for ERI. She is supervising many Graduate Projects from 2006 till now. She is also a technical member at both the ERI projects committee and at the telecommunication networks committee.