ISSN (ONLINE): 2349-7084
GLOBAL IMPACT FACTOR 0.238
DIIF 0.876

INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING IN RESEARCH TRENDS
VOLUME 2, ISSUE 6, JUNE 2015, PP 387-391

Investigation on Secondary Memory Management in Wireless Sensor Network

J.David Sukeerthi Kumar

Abstract: This paper presents a survey on Importance of Secondary Memory management in Wireless Sensor Network (WSN) Operating Systems (OSs) .Sensor nodes may also have a storage unit or debugging unit. The storage unit is an external memory device that works as a secondary memory, keeping a data log from an energy perspective. In recent years, WSNs have received tremendous attention in the research community, with applications in battlefields, industrial process monitoring, home automation, and environmental monitoring. A WSN is a huge dynamic network in this aspect a severe environmental circumstances and low power of battery nodes might be die. Furthermore, a WSN is composed of limited memory and computational abilities. WSNs invariably operate in an unattended mode and in many scenarios it is impossible to replace sensor nodes after deployment, therefore a fundamental objective is to optimize the sensor nodes life time. These characteristics of WSNs impose additional challenges on OS design for WSN, and consequently, OS design for WSN deviates from traditional OS design. The purpose of this survey is to highlight major concerns pertaining to OS design in WSNs and to point out strengths and weaknesses of contemporary OSs for WSNs, keeping in mind the requirements of emerging WSN applications.

Keywords: Wireless Sensor Network, Secondary Memory management, TinyOS, Dynamic Memory Location

1. INTRODUCTION

A sensor node, also known as a mote (chiefly in North America), is a node in a sensor network that is capable of performing some processing, gathering sensory information and communicating with other connected nodes in the network. A mote is a node but a node is not always a mote. Infrastructural support for WSN applications in the form of operating systems is becoming increasingly important. It bridges the gap between hardware simplicity and application complexity, and it plays a central role in building scalable distributed applications that are efficient and reliable. One of the important OS design issues which requires lot more attention is memory management in WSN[4].

Memory in current sensor nodes consists of: RAM (for fast data storage), internal flash (for code storage), EEPROM (for data storage), and external flash which is required for data persistence.

In a traditional operating system, memory management refers to various techniques used for allocation and de-allocation of memory blocks to different processes and threads. Commonly used memory management techniques are static memory allocation & dynamic memory allocation. Earlier, very little or no support used to be provided for managing the memory assuming that only single application runs on the sensor node.

• ***J.David Sukeerthi Kumar**,. Currently he is working as Assistant Professor in Department of Computer Science Engineering in Currently he is working as Assistant Professor in Department of Computer Science and Engineering Santhiram Engineering College, Nandyal.Kurnool ,AndhraPradesh.India*

But with the emergence of new application domains for WSNs which support real time traffic, multithreaded, multi core designs, multimedia streams of data for transfer, these WSNs provide the mechanism for concurrent execution of multiple threads [4]. Since the memory is one of the Constrained resources in case of WSN, it becomes a challenging task for applying these memory management techniques efficiently to various processes & threads with real time traffic.

2. COMPONENTS OF SENSOR NODES

The main components of a sensor node are a microcontroller, transceiver, external memory, power source and one or more sensors.

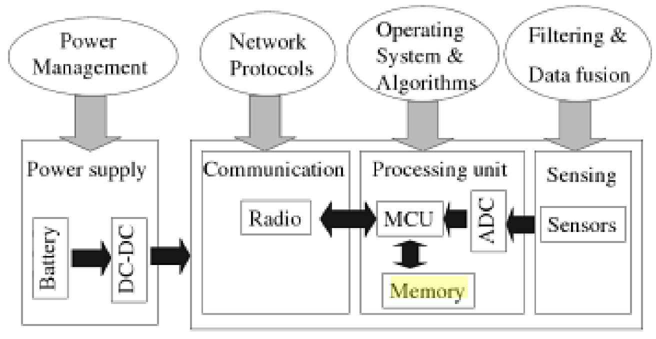


Fig 1.Block Diagram of Sensor node

ISSN (ONLINE): 2349-7084

INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING IN RESEARCH TRENDS
VOLUME 2, ISSUE 6, JUNE 2015, PP 387-391

Main components of a WSN node Controller Communication device(s) Sensors/actuators Memory Power supply.

CONTROLLER: The controller performs tasks, processes data and controls the functionality of other components in the sensor node.

MAIN OPTIONS Microcontroller – general purpose processor, optimized for embedded applications, low power consumption DSPs – optimized for signal processing tasks, not suitable here FPGAs – may be good for testing ASICs – only when peak performance is needed, no flexibility

TASKS Execution of (time sensitive) tasks Control of communication protocols Execution and control of the primary application program Energy management multiple operation modes (active, idle, listen, sleep,)

Controller	Atmega128	TI MSP430	PIC 18F720	ARM920T
RAM	4 KB	10 KB	4 KB	512 KB
Flash	128 KB Program 512 KB Serial	48 KB	128 KB Program 512 KB Filesystem 1KB EEPROM	4 MB
Clock rate	8 MHz	8 MHz	20 MHz	180 MHz
Energy consumption	Active: 8 mA Sleep: <15 uA	Active: 1.8 mA Sleep: 5.1 uA	Active: 8 mA	Active: 25-90 mA Sleep: 32 uA
Plattform	MicaZ	TelosB	Particle	SunSPOT
Costs	~ 12 Euro		~ 5 Euro	

COMMUNICATION DEVICE(S)

Sensor nodes often make use of ISM band, which gives free radio, spectrum allocation and global availability. The possible choices of wireless transmission media are radio frequency (RF), optical communication (laser) and infrared. Lasers require less energy, but need line-of-sight for communication and are sensitive to atmospheric conditions. Infrared, like lasers, needs no antenna but it is limited in its broadcasting capacity. Radio frequency-based communication is the most relevant that fits most of the WSN applications. WSNs tend to use license-free communication frequencies: 173, 433, 868, and 915 MHz; and 2.4 GHz. The functionality of both transmitter and receiver are combined into a single device known as a transceiver. Transceivers often lack unique identifiers. The operational states are transmit, receive, idle, and sleep. Current generation transceivers have built-in state machines that perform some operations automatically.

Most transceivers operating in idle mode have a power consumption almost equal to the power consumed in receive mode.[6] Thus, it is better to completely shut down the transceiver rather than leave it in the idle mode when it is not transmitting or receiving. A significant amount of power is consumed when switching from sleep mode to transmit mode in order to transmit a packet.

TRANSCEIVER STATES

TRANSCEIVERS CAN BE PUT INTO DIFFERENT OPERATIONAL STATES, TYPICALLY:

• Transmit

• Receive

• Idle – ready to receive, but not doing so

SOME FUNCTIONS IN HARDWARE CAN BE SWITCHED OFF, REDUCING ENERGY CONSUMPTION A LITTLE

• Sleep – significant parts of the transceiver are switched off • Not able to immediately receive something

• Recovery time and startup energy to leave sleep state can be significant

EXTERNAL MEMORY

Sensor nodes may also have a storage unit or debugging unit. The storage unit is an external memory device that works as a secondary memory, keeping a data log from an energy perspective, the most relevant kinds of memory are the on-chip memory of a microcontroller and Flash memory—off-chip RAM is rarely, if ever, used. Flash memories are used due to their cost and storage capacity. Memory requirements are very much application dependent. Two categories of memory based on the purpose of storage are: user memory used for storing application related or personal data, and program memory used for programming the device. Program memory also contains identification data of the device if present.

POWER SOURCE

A wireless sensor node is a popular solution when it is difficult or impossible to run a mains supply to the sensor node. However, since the wireless sensor node is often placed in a hard-to-reach location, changing the battery regularly can be costly and inconvenient. An important aspect in the development of a wireless sensor node is ensuring that there is always adequate energy available to power the system. The sensor node consumes power for sensing, communicating and data processing. More energy is required for data communication than any other process. The energy cost of transmitting 1 Kb a distance of 100 metres (330 ft) is approximately the same as that used for the execution of 3 million instructions by a 100 million instructions per second/W processor. Power is stored either in batteries or capacitors. Batteries, both rechargeable and non-rechargeable, are the main source of power supply for sensor nodes. They are also classified according to electrochemical material used for the electrodes such as NiCd (nickel-cadmium), NiZn(nickel-zinc), NiMH (nickel-metal hydride), and lithium-ion. Current sensors are able to renew their energy from solar sources, temperature differences, or vibration. Two power saving policies used are Dynamic Power Management (DPM) and Dynamic Voltage Scaling (DVS).[7] DPM conserves power by shutting down parts of the sensor node which are not currently used or active. A DVS scheme varies the power levels within the sensor node depending on the non-deterministic workload. By varying the voltage along with the frequency, it is possible to obtain quadratic reduction in power consumption.

Primary batteries			
Chemistry	Zinc-air	Lithium	Alkaline
Energy (J/cm^3)	3780	2880	1200
Secondary batteries			
Chemistry	Lithium	NiMHd	NiCd
Energy (J/cm^3)	1080	860	650

Table 1: Energy per volume (Joule per cubic centimeter):

Energy source	Energy density
Batteries (zinc-air)	$1050 - 1560\,\mathrm{mWh/cm^3}$
Batteries (rechargable lithium)	$300\,\mathrm{mWh/cm^3}$ (at $3 - 4\,\mathrm{V}$)
Energy source	**Power density**
Solar (outdoors)	$15\,\mathrm{mW/cm^2}$ (direct sun)
	$0.15\,\mathrm{mW/cm^2}$ (cloudy day)
Solar (indoors)	$0.006\,\mathrm{mW/cm^2}$ (standard office desk)
	$0.57\,\mathrm{mW/cm^2}$ (< 60 W desk lamp)
Vibrations	$0.01 - 0.1\,\mathrm{mW/cm^3}$
Acoustic noise	$3 \cdot 10^{-6}\mathrm{mW/cm^2}$ at $75\,\mathrm{Db}$
	$9,6 \cdot 10^{-4}\mathrm{mW/cm^2}$ at $100\,\mathrm{Db}$
Passive human-powered systems	$1.8\,\mathrm{mW}$ (shoe inserts)
Nuclear reaction	$80\,\mathrm{mW/cm^3}$, $10^6\,\mathrm{mWh/cm^3}$

Table 2. Energy scavenging – overview

SENSORS

Sensors are hardware devices that produce a measurable response to a change in a physical condition like temperature or pressure. Sensors measure physical data of the parameter to be monitored. The continual analog signal produced by the sensors is digitized by an analog-to-digital converter and sent to controllers for further processing. A sensor node should be small in size, consume extremely low energy, operate in high volumetric densities, be autonomous and operate unattended, and be adaptive to the environment. As wireless sensor nodes are typically very small electronic devices, they can only be equipped with a limited power source of less than 0.5-2 ampere-hour and 1.2-3.7 volts.

Sensors are classified into three categories: passive, omni-directional sensors; passive, narrow-beam sensors; and active sensors. Passive sensors sense the data without actually manipulating the environment by active probing. They are self powered; that is, energy is needed only to amplify their analog signal. Active sensors actively probe the environment, for example, a sonar or radar sensor, and they require continuous energy from a power source. Narrow-beam sensors have a well-defined notion of direction of measurement, similar to a camera. Omni-directional sensors have no notion of direction involved in their measurements.

The overall theoretical work on WSNs works with passive, omni-directional sensors. Each sensor node has a certain area of coverage for which it can reliably and accurately report the particular quantity that it is observing. Several sources of power consumption in sensors are: signal sampling and conversion of physical signals to electrical ones, signal conditioning, and analog-to-digital conversion. Spatial density of sensor nodes in the field may be as high as 20 nodes per cubic meter.

3. COMPARATIVE ANALYSES

- Due to the multi-threaded semantics, every Mantis program must have stack space allocated from the system heap, and locking mechanisms must be used to achieve mutual exclusion of shared variables. In contrast, Contiki uses an event based scheduler without preemption, thus avoiding allocation of multiple stacks and locking mechanisms. Preemptive multi-threading is provided by a library that can be linked with programs that explicitly require it.

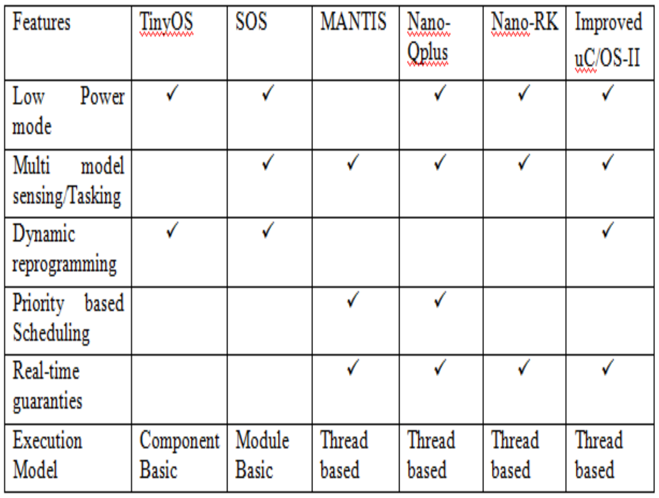
- Contiki's event kernel is significantly larger than that of TinyOS because of the different services provided. While the TinyOS event kernel only provides a FIFO event queue scheduler, the Contiki kernel supports both FIFO events and poll handlers with priorities.

- The experimental results show that MANTIS is more predictable than TinyOS. Specifically, the packet forwarding task execution time in MANTIS has a low variation and is independent of other activity such as the sensing task. Thus, MANTIS would be preferable in situations that need deterministic and timely processing. However, as the experiments show, the MANTIS system is not as power-efficient as TinyOS. Thus, TinyOS would seem preferable if energy consumption is deemed to be of primary importance.

- The dynamic loading mechanism of LiteOS follows the line of several previous efforts that did not involve virtual memory such as TinyOS (using XNP), SOS, and Contiki. Lalit Saraswat et al Int J Engg Techsci Vol 1(1) 2010,41-47 IJETS|www.techsciencepub.com/ijets 45

- TinyOS' latency is much smaller than the others because TinyOS' task creation simply means assigning function pointer of a task to a ready queue. It does not need memory to be allocated or copied because TinyOS' scheduler is FIFO (non-preemptive). However, MANTIS and NanoQplus[8] operating systems requires memory allocation of task control block.

- Nano-RK[5] supports power management techniques and provides several power-aware APIs for system use. While low-footprint operating systems such as μC/OS support real-time scheduling, they do not have support for wireless networking.
- The Nano-RK is the most closely related work to PAVENET OS. Nano-RK is a preemptive multitask operating system supporting real-time tasks. Additionally, Nano-RK is more portable than PAVENET OS. However, Nano-RK has more context switch overhead than PAVENET OS because Nano-RK has to preserve CPU context by software. Nano-RK needs several dozens of μs for task switching whereas PAVENET OS needs several μs.
- Like PAVENET OS, MANTIS is a thread model operating system. The difference between MANTIS and PAVENET OS is the implementation of the thread model. MANTIS uses time-sliced multithreading, whereas the threading of PAVENET OS is not time-sliced.
- TinyOS can port to PAVENET modules, but PAVENET OS cannot port to MICA2[3] because of its CPU specific architecture.



Features	TinyOS	SOS	MANTIS	Nano-Qplus	Nano-RK	Improved uC/OS-II
Low Power mode	✓	✓		✓	✓	✓
Multi model sensing/Tasking		✓	✓	✓	✓	✓
Dynamic reprogramming	✓	✓				✓
Priority based Scheduling			✓	✓		
Real-time guaranties			✓	✓	✓	✓
Execution Model	Component Basic	Module Basic	Thread based	Thread based	Thread based	Thread based

DESIGN CHALLENGES OF WSN

✓ Typically, severely energy constrained. •

● Limited energy sources (e.g., batteries). •

• Trade-off between performance and lifetime.

✓ Self-organizing and self-healing. •

● Remote deployments. ˆ

✓ Scalable. •

• Arbitrarily large number of nodes

✓ Heterogeneity. •

● Devices with varied capabilities. •

● Different sensor modalities. •

• Hierarchical deployments. ˆ

● Adaptability. •

✓ Adjust to operating conditions and changes in application requirements.

✓ Security and privacy. •

● Potentially sensitive information. •

• Hostile environments.

4. MEMORY MANAGEMENT ISSUES &CHALLENGES

There are many issues & challenges that should be considered while designing efficient memory management system. In this section, we will see some of the major concerns in this area.

VIRTUAL MEMORY

Many of the sensor nodes lack or have very limited support for the address translation (MMU). As it is power intensive operation & sensor nodes have very limited power & storage capabilities, it is really challenging to provide more memory to the applications than assigned by the physical memory. Especially the work done in virtual memory management for WSN is very limited.[2]

SECONDARY STORAGE MANAGEMENT

As many emerging WSN applications require more memory, and these applications require management of large databases & real time traffic, the need for secondary storage increases. In these types of applications, data must be stored in the network, and thus storage becomes a primary resource which, in addition to energy, determines the useful lifetime and coverage of the network. There are only few OSs that provide a file system to manage secondary storage. So the scalable(distributed) file system for WSNs to manage secondary storage has to be designed for such applications. The collaborative storage provides more suitability to meet the goals of storage management.

SMALL FOOTPRINT

The storage capacity available on a sensor node is in terms of few kilobytes due to which the OS has to be designed with a very small footprint [8]. It is a fundamental characteristic of a sensor Operating System.

TASK SCHEDULING & RESOURCE SHARING

It provides the task environment for executing long running application. The task scheduling can be either event-driven or multi-threaded. Here also memory management should be done in efficient way for memory allocation between scheduled tasks. Another issue to be considered is the resource sharing between executions of multiple applications. For this, the efficient concurrency control & memory protection mechanism should be provided between these applications.

DYNAMIC MEMORY ALLOCATION

Data memory has been a very scarce resource in sensor networks [2]. Thus, its efficient utilization is necessary. Allocation of a memory to the dynamic data structures becomes a challenging task on the sensor node as the memory requirement varies depending on the size of the data structure. WSN applications with increasing application domains require efficient dynamic memory allocation techniques to be designed.

REPROGRAMMING & MEMORY MANAGEMENT

Reprogramming & up-gradation of software's on already deployed nodes is challenging because of the fact that sensor networks may be deployed in physically unattended environment and often consist of few thousands of nodes. Therefore, an already deployed sensor network must be wirelessly reprogrammable irrespective of above problems [2].

Reprogramming requires dynamic loading & unloading of the software modules or individual services & proper memory management policies like contiguous memory allocation, de-allocating memory, and paging [3]. For these policies to be enforced, proper APIs should be provided by OS to support reprogramming.

5. FUTURE SCOPE

In this section, we will discuss some of the challenges where further research work is required in this area: As new application areas with real time traffic need more memory, the sensor nodes with large secondary storage are required [6]. As huge amount of data is collected & processed at the nodes, this data has to be stored & maintained in large databases. So the requirement of secondary memory management here is also increased. To achieve this, much more improvement in file system to manage the secondary storage & large databases is required.

6. CONCLUSION

In this paper we examine Sensor node components with various aspects and also investigate various memory management issues and challenges in this regard examine comparative analysis of various WSN OS. Describing Design Challenges of WSN further investigations are required to give stronger real time support & efficient memory management techniques for WSNs. Much more attention is required for resolving the problems in areas like WMSNs, integration of WSN with cloud computing etc.

REFERENCES:

[1]. Akyildiz, I.F.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E. Wireless Sensor Networks: A Survey. Comput. Netw. 2002, 38, 393-422.

[2] http://www.comp.nus.edu.sg/~doddaven/cata.pdf

[3]. Levis, P.; Madden, S.; Polastre, J.; Szewczyk, R.; Whitehouse, K.; Woo, A.; Gay, D.; Hill, J.; Welsh, M.; Brewer, E.; Culler, D. Tinyos: An Operating System for Sensor Networks; Available online: http://dx.doi.org/10.1007/3-540-27139-2_7 (accessed on 17 April 2011).

[4]. Akyildiz, I.F.; Melodia, T.; Chowdhury, K.R. A Survey on Wireless Multimedia Sensor Networks. Comput. Netw. 2007, 51, 921-960.

[5]. Kim, H.; Cha, H. Multithreading Optimization Techniques for Sensor Network Operating Systems. In Proceedings of the 4th European Conference on Wireless Sensor Networks, Delft, The Netherlands, January 2007; pp. 293-308.

[6]. Klues, K.; Liang, C.J.M.; Paek, J.; Musaloiu, R.; Levis, P.; Terzis, A.; Govindan, R. TOSThread: Thread-Safe and Non-Invasive Preemption in TinyOS. In Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, Berkeley, CA, USA, 4–6 November 2009; pp. 127-140.

[7]. Cooprider, N.; Archer, W.; Eide, E.; Gay, D.; Regehr, J. Efficient Memory Safety for Tinyos. In Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (SenSys'07), New York, NY, USA, November 2007; pp. 205-218.
[8].http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5462978&url=http%3A%2F%2Fieeexplore.ieee.org

ABOUT THE AUTHOR

J.David Sukeerthi Kumar , Received B.Tech degree from Narayana Engineering college, Nellore (JNTU Hyd) , M.Tech from Rajeev Gandhi Memorial college of Engineering and Technology (RGMCET),Nandyal (JNTU Hyd). Currently he is working as Assistant Professor in Department of Computer Science and Engineering Santhiram Engineering College, Nandyal.