REAL-TIME SCHEDULING ALGORITHMS FOR WIRELESS SENSOR NETWORK

D.G.HARKUT1, M. S. ALI2 AND POONAM LOHIYA3

1H.O.D, PRMCEAM, BADNERA, 2PRINCIPAL, PRMCEAM, BADNERA AND 3M.E 2NDYR, PRMCEAM, BADNERA

ABSTRACT

In Recent era, researchers attention is get magnetize towards a widely used technology known as a wireless sensor networks (WSNs) used in the many of applications related to variety of fields including military, healthcare monitoring, biological, home, vehicular monitoring, infrastructure monitoring, building energy monitoring and industrial sensing. Many of applications in WSNs are real time applications that are requested to run in real time way. To support such real time applications we need Real Time Operating System (RTOS) which provides logically correct results and also deadline has to be met. This paper presents an overview of existing scheduling algorithms which helps to schedule tasks in real time systems. Next, we discuss work in sensornet operating system (OS) design. Then, the specialties what sensornet OS should posses are discussed in detail. At last, we proposed Micro Controller OS-II (µC/OS-II) with Earliest Deadline First (EDF) algorithm.

KEYWORDS

Wireless Sensor Network, Real Time Operating System, µC/OS-II operating system, Earliest Deadline First.

1. INTRODUCTION

A WSN composed of number of wireless interconnected sensors. These are usually tiny autonomous devices that are used to monitor some physical conditions or other values with their sensors and use short range wireless link for communication between them and between higher level systems. The measured values typically are temperature, humidity, light intensity; moisture etc.WSN is found lot of interest due to great variety of applications like environmental monitoring, military surveillance, biomedical systems, intelligent parking, healthcare applications and industrial applications. Most of these applications related with the real world environments and the delivery of data is limited to certain timing restraints [11]. It is a special type of real time embedded systems where deadline is one of the critical parameter. Applications in embedded systems are usually domain-specific. Most of the applications will require different operating systems platform to provide various services. The basic functionalities of an OS include resource abstraction, process management, memory management, interrupt management and device management, scheduling policies, multithreading, and multitasking [1]. The OS also have a very efficient inter-thread communication subsystem so that if a thread wants to communicate or synchronize with each other, it should be able to do so without fail [18]. Over the past years we have observed that OS plays an important role in building scalable distributed applications that are efficient and dependable. WSNs typically does not use general purpose operating system rather they use OS designed directly for them called special purpose OS i.e. RTOS. RTOS supports applications that meet deadlines and certain timing constraints to providing logically correct results [17]. To meet the deadlines in Real time system we need to schedule the task properly so that no one process goes under starvation or miss the deadline, for that different scheduling techniques were used. The main objective of a real-time task scheduler is to meet the deadline of tasks in the system. Mostly all the real-time systems in existence use multitasking and pre-emption.

µC/OS-II is a real-time pre-emptive multitasking embedded OS kernel. A pre-emptive kernel is used when system responsiveness is important; therefore, µC/OS-II and most commercials real-time kernels are pre-emptive. µC/OS-II is a completely portable, ROMable, scalable, real-time kernel. It provides services like task management, memory management and time management. µC/OS-II kernel supports pre-emptive algorithms such as round-robin, rate monotonic scheduling policy [29].

In this paper, we proposed µC/OS-II operating system for wireless networked sensors with suitable scheduling policy. The remainder of this paper is organized as follows: Section 2 introduces related works of sensor network OS with suitable scheduling policy. Section 3 elaborates the task scheduling algorithms required to schedule the task in WSN. Section 4 discusses the specialties which a sensor OS should possess. Finally, section 5 concludes this paper with some new ideas for future study.

2. RELATED WORK

The huge potential of WSN applications needs the RTOS to be suitable for different operating environments, from a simple single-task event to a real-time multi-thread system. It means that the RTOS must be resource-and context aware to minimize energy consumption. This section presents a past work that has been done on embedded OS and also discussed scheduling policies.

In [14], Levis et al. proposed Berkeley's TinyOS architecture which is designed for WSN. TinyOS is a well-known operating system having light weight, low power and the *Mote* platform has been widely used in many kinds of applications. It is currently a fundamental framework of research on wireless sensor networks. The OS has a footprint that fits in 400 bytes and having monolithic kernel. Since TinyOS kernel supports FIFO scheduling policy but FIFO scheduling policy having some disadvantages that are also associated with the TinyOS scheduler. TinyOS does not support real-time application; hence this OS is not a good choice to run real time applications.

In [6], Dunkels et al. proposed ContikiOS as a lightweight and flexible OS for tiny network sensors. It is having event-driven architecture written in C for WSN sensor nodes. It does not employ any special kind of scheduling algorithm for real time applications. Events are classified as synchronous and asynchronous and they are scheduled as they arrive. In case of interrupts, interrupt handlers of an application runs with respect to their priority.

In [3], Bhatti et al. proposed the Mantis OS is energy efficient, multithreaded OS provides scheduler which uses round robin scheduling policy within the each priority class. This policy means the highest priority thread class can cause starve to lower priority thread. As compare to TinyOS or Contiki scheduler, the Mantis OS scheduler is better because of pre-emptive priority scheduling technique that may support real-time task. But still a better scheduling policy is required to support real-time tasks.

In [7], Eswaran et al. proposed Nano-RK a reservation-based energy-aware embedded OS having small footprint with networking support. Nano-RK OS do not support end-to-end deadlines guarantees associated with packet delivery. It provides only static scheduling. To support real-time applications, Nano-RK uses rate monotonic scheduling algorithm which statically assigns priorities to tasks.

In [4] [15], Cao Q et al. LiteOS, a multithreaded, Unix-like abstractions for sensor network. This OS is more familiar to user. This OS schedule the real-time tasks with the help of round robin algorithm. Because of some disadvantages of round robin algorithm we need a better scheduling algorithm; therefore LiteOS is unable to support real-time applications.

Above discussion clarifies that some of the OSs provides the services for real-time application with a priority based scheduling algorithms while others do not support for this.

3. Real-time Scheduling Algorithm

Most of the tasks in WSN are requested to run in a real-time way. A system is said to be a real-time system if it generates a result within a specified time so that job will complete its execution before deadline and system will never fail if all jobs will complete its task before deadline. The real time systems accepts commands from external peripherals, processes the data and then perform desired action. Mostly we can divide real-time system into two main categories: Hard real-time and soft real-time system. If a system is a hard real time then all task must be completed before its deadline so that system will never face any catastrophic situation whereas in soft real time system, if task miss their deadline then no such disastrous situation arise. As real-time systems execute critical tasks, therefore it must be designed very carefully. For that, many scheduling policy has been already designed [28].

A fundamental operation of an OS is scheduling the task. In order to meet a program's temporal requirements of real-time systems, it is of the utmost importance that the scheduling algorithm should produce a predictable schedule, that is, at all times it should be known that which task is going to be executed [9]. Figure 1 shows the basic scheduling algorithm function.

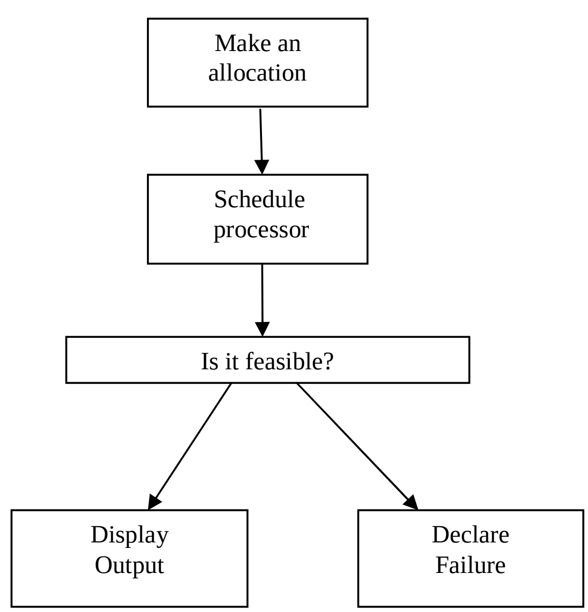

Figure 1. Basic Function of Scheduling Algorithm

3.1 Different available scheduling algorithms and their characteristics

In Real-time systems scheduling algorithms are classified into two categories: Static algorithm and Dynamic algorithm. Based on execution attributes of tasks, dynamic algorithm assigns priorities at runtime. This algorithm allows switching of priorities between tasks. In contrast with dynamic algorithm, a static algorithm assigns priorities at design time. All assigned priorities remain fixed throughout the execution of task. Figure 2 gives the classification of available scheduling algorithms for real-time systems [13] [16].

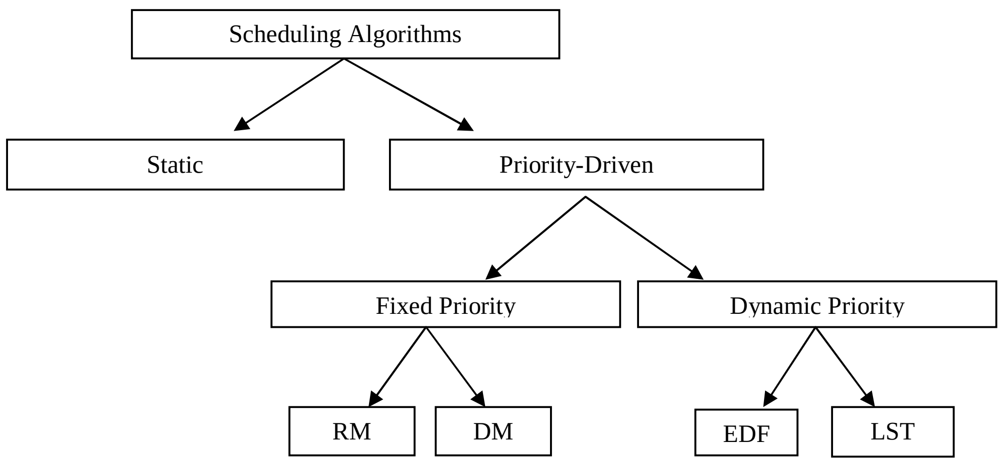

Figure 2. Types of Real time Scheduling Algorithms

Rate Monotonic (RM): In RM algorithm tasks have to be periodic in nature and deadline must be equal to its period. Tasks are scheduled according to their period. The rate of task is the inverse of its period. This algorithm implemented by assigning fixed priority to tasks, the higher its rate, higher the priority.

Deadline Monotonic (DM): The other algorithm for scheduling all the real time tasks based on their deadline is known as deadline monotonic. In this algorithm priority are decided by considering relative deadline. If one task will get higher priority than the other then its relative deadline must be shorter as compared to other tasks. RM and DM are identical except priorities are automatically computed from rate of task or deadline.

Least Slack Time First (LST): It is type of dynamic algorithm which assigns priority dynamically. Tasks are scheduled according to their slack: the smaller the slack, the higher the priority. Slack is computed by using the difference between the deadline, the ready time and the run time.

Earliest Deadline First (EDF): The most common dynamic priority scheduling algorithm for real-time systems is the EDF. Here priorities are dynamically reassigned at run-time based on the time still available for each task to reach its next deadline. Both static and dynamic systems are scheduled by EDF algorithm.

Maintain a queue of tasks in ascending order of deadline so that whenever a processor gets free then by using EDF scheduling policy task of the head of the queue will be assigned to the processor. When new task arrives, its deadline will be compared with the deadline of currently executing task, and in case if deadline of newly arrived task is closer to the current time, it will receive the processor and the old task will be pre-empted and placed back in the queue. The EDF algorithm is optimal compared with the other real-time algorithms and if task is not scheduled by EDF then all other algorithms will fail to schedule that task [8] [19] [22]. As compared to other algorithms EDF is simple to implement and gives much better utilization of processor.

In WSN, most of the tasks are appeared dynamically over the network. With the help of available scheduling techniques like RM, DM, LST the most of the task are not scheduled before the deadline and also this algorithms are responsible for data loss, overload and decline of throughput. In most applications, the life of sensor nodes depends on their battery support. The sensor node has limited battery energy and thus the sensor node must use available resources effectively and manage the energy to extend the lifetime of the network as much as it can. Therefore managing energy is very difficult problem while building a WSN.

This paper proposes an EDF scheduling algorithm which helps to schedule tasks dynamically and enhance the throughput, reduce the overload and also helps to decrease energy consume in data transmission and routing [22][27]. EDF improves the system's performance and allows a better exploitation of resources.

4. SENSOR OS DESIGN AND CHALLENGES

In this section we have discussed the challenges while designing a efficient, reliable OS for WSN.

4.1 Small Footprint. The current era of embedded processors demanding larger Read Only Memory sizes with smaller Random Access Memory sizes [29]. Due to the limitation of memory on a sensor node demands the OS to be designed with a very small footprint. It is a fundamental characteristic of a sensor network OS and is the primary reason why so many sophisticated embedded OS cannot be easily ported to sensor nodes [26].

4.2 Power management capability. Energy is important parameter in WSNs that must achieve long lifetimes while operating on battery energy. Sensor nodes provide very limited battery life-time. Thus possessing power management capability is essential, which helps to improve system performance and extend the battery lifetime [5].

4.3 Portability. In WSN the hardware platforms are evolving day-by-day. Portability is considered to be an important issue as everyone is working on their customized hardware platforms. The OS should be designed in such a way that it is easily portable to different hardware platforms with minimal changes.

4.4 Real-Time Guarantee. As most applications in WSN are time-sensitive in nature where data must be forwarded and relayed on a timely basis, real-time guarantee is a necessary requirement for such applications. By using real-time scheduler with proper scheduling technique real-time constraints of the application can be satisfied [10].

4.5 Networking Stack. The networking stack facilitates developing distributed WSN applications. The OS should support multi-hop wireless networking, routing. It also handles reliable packet transmission, multicasting, queue management, radio chip configuration, and Medium Access Control (MAC) [26].

4.6 Programming Convenience. For application programmers OS should provide a convenient programming environment. Many Sensor network applications are diverse and demanding. Hence importance on development of sensor network applications programming convenience is of great importance.

4.7 Dynamic Reprogramming. Dynamic reprogramming is an especially useful feature for wireless networked sensors. It is the process of updating the software dynamically running on the sensor nodes. Without reprogramming, it is difficult to perform operations like modification, deletion or adding the contents in the software from the running system in WSN [23].

4.8 Customizability. Mostly all the software platforms developed for WSN is application specific. Different applications demand different services from operating system [1]. These requirements may be small footprint, real-time guarantees, reprogramming. The design of OS should be in such a way that it should be easily customizable and extensible to various applications.

4.9 Timeliness and Schedulability. Most sensor applications tend to be time-sensitive in nature where processing must be completed within the defined time-bound otherwise system will fail. Scheduling is technique for allocating tasks on processors to ensure that deadlines are to meet. We need RTOS to support this scheduling technique.

Following table represents the summary of OS which supports the above mentioned challenges.

Table 1 Summary of OS

Features	**Tiny OS**	**Contiki OS**	**Mantis OS**	**Nano-Rk OS**	**Lite OS**	**µC/OS-II**
Priority based scheduling	No	No	No	Yes	Yes	Yes
Real time guarantee	No	No	No	Yes	No	Yes
Dynamic Reprogramming	No	Yes	Yes	Yes	Yes	Yes
Memory Management	No	No	No	No	Yes	Yes
Low Power Mode	Yes	Yes	Yes	Yes	Yes	Yes
Scheduling algorithms used	FIFO	Run time scheduling	Round Robin	Rate Monotonic	Round Robin	Rate Monotonic

5. CONCLUSION AND FUTURE WORK

In this paper, we provide an overview of existing work; discuss the various scheduling algorithms with their characteristics and also proposed EDF scheduling algorithm with their benefits over the other real time scheduling algorithms. Next we discuss the challenges of in the OS design space. In this paper, we discussed the µC/OS-II operating system with its features. µC/OS-II operating system supports various real time applications in WSN. Currently, we are designing and implementing EDF algorithm for scheduling the entire tasks in WSN by using µC/OS-II RTOS. Further study is required to improve performance.

REFERENCES

[1] Adi Mallikarjuna Reddy V AVU Phani Kumar, D Janakiram, and G Ashok Kumar, (2007). Operating Systems for Wireless Sensor Networks: A Survey Technical Report.

[2] Andre Rodrigues, Tiago Camilo, Jorge Sa Silva, Fernando Boavida, (2012). Diagnostic Tools for Wireless Sensor Networks: A Comparative Survey, Springer Science Business Media, LLC.

[3] Bhatti S.; Carlson J.; Dai H.; Deng J.; Rose J.; Sheth A.; Shucker B.; Gruenwald C.; Torgerson H.R., (2005). Mantis OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms. Mobile. Network, 563-579.

[4] Cao, Q.; Abdelzaher, T.; Stankovic, J.; He, T., (2008). 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 2008), St. Louis, MO, USA, 22–24.

[5] Chi-Tsun Cheng, Chi K. Tse, and Francis C. M. Lau, (2010). An Energy-Aware Scheduling Scheme for Wireless Sensor Networks, IEEE transaction on vehicular technology, vol. 59, no. 7.

[6] Dunkels, A.; Gronvall, B.; Voigt, T., (2004). Contiki a Lightweight and Flexible Operating System for Tiny Networked Sensors. In Proceedings of the 9th Annual IEEE International Conference on Local Computer Networks, Washington, DC, USA; pp. 455-462.

[7] Eswaran, A.; Rowe, A.; Rajkumar, R., (2005). Nano-RK: An Energy-Aware Resource-Centric RTOS for Sensor Networks. In Proceedings of the 26th IEEE Real-Time Systems Symposium, Miami, FL, USA, 5–8.

[8] Fengxiang Zhang, Alan Burns, (2009). Schedulability Analysis for Real-Time Systems with EDF Scheduling IEEE transactions on computers, vol. 58.

[9] Hai-ying Zhou, Feng Wu, Kun-mean Hou, (2008). An Event-driven Multi-threading Real-time Operating System dedicated to Wireless Sensor Networks. The 2008 International Conference on Embedded Software and Systems (ICESS2008) IEEE.

[10] Jane W.S. Liu, (2001). Real-Time Systems, Pearson Education, India, pp. 121 & 26.

[11] Jennifer Yick, Biswanath Mukherjee, Dipak Ghosal, (2008).Wireless sensor network survey, Elsevier B.V. Computer Networks.

[12] Kathleen Baynes, Chris Collins, Eric Fiterman, Brinda Ganesh, Paul Kohout, Christine Smit, Tiebing Zhang, and Bruce Jacob,(2003). The Performance and Energy Consumption of Embedded Real-Time Operating Systems,IEEE transactions on computers, vol. 52, no. 11.

[13] Kayvan Atefi, Mohammad Sadeghi, Arash Atefi,(2011).Real-Time Scheduling Strategy for Wireless Sensor Networks O.S. International Journal of Distributed and Parallel Systems (IJDPS) vol.2, no.6.

[14] Levis, P.,Madden, S. Polastre, J., Szewczyk, R.,Whitehouse, K.Woo, A. Gay, D. Hill, J. Welsh, M.Brewer, E. Culler, D.,(2011). Tinyos: An Operating System for Sensor Networks.

[15] LiteOS. LiteOS [online], (2011)[cit.2012-05-28]. Available from: http://www.liteos.net

[16] M.Kaladevi and Dr.S.Sathiyabama, (2010). A Comparative Study of Scheduling Algorithms for Real Time Task. International Journal of Advances in Science and Technology, vol. 1, no. 4.

[17] M.V. Panduranga Rao, K.C. Shet, R.Balakrishna, K. Roopa, (2008). Development of Scheduler for Real Time and Embedded System Domain, 22nd International Conference on Advanced Information Networking and Applications – Workshops, IEEE Computer Society, pp. 1-6.

[18] Octav Chipara, Chenyang Lu, and Gruia-Catalin Roman, (2013). Real-Time Query Scheduling for Wireless Sensor Networks,IEEE transactions on computers, vol. 62, no. 9.

[19] Pinkesh Pachchigar, P.Eswaran, Amol Kashinath Boke, (2013). Design and Implementation of Deadline based EDF Algorithm on ARM LPC2148, Proceedings of 2013 IEEE Conference on Information and Communication Technologies (ICT 2013), pp. 994-997.

[20] Ranjan Dasgupta, (2008). Anatomy of RTOS and Analyze the Best-Fit for Small, Medium and Large Footprint Embedded Devices in Wireless Sensor Network, The Second International Conference on Sensor Technologies and Applications, IEEE Computer Society, pp. 598-603.

[21] Rowe, A.; Lakshmanan, K.; Yhu, H.; Rajkumar, R., (2008). Rate-Harmonized Scheduling for Saving Energy. In Proceedings of the 29th IEEE Real-Time Systems Symposium, Barcelona, Spain.

[22] Rym Chéour, Sébastien Bilavarn, Mohamed Abid, (2011).Exploitation of the EDF Scheduling in the Wireless Sensors Networks, International Journal of Measurement Technologies and Instrumentation Engineering, 1(2), 14-27.

[23] Sangho Yi, Hong Min, Junyoung Heo, Boncheol Gu, Yookun Cho, Jiman Hong, Jinwon Kim, Kwangyong Lee, and Seungmin Park, (2006). Performance Analysis of Task Schedulers in Operating Systems for Wireless Sensor Networks M. Gavrilova et al. (Eds.): ICCSA 2006, LNCS 3983, Springer-Verlag Berlin Heidelberg, pp. 499–508.

[24] TinyOS[EB/OL], http:// www.tinyos.net, 2007-6-1.

[25] TinyOS Network Working Group; Tutorials#Network_Protocols (accessed on 17 April 2011).

[26] Wei Dong, Chun Chen, Xue Liu, Jiajun Bu, (2010). Providing OS Support for Wireless Sensor Networks: Challenges and Approaches, IEEE Communications Surveys & Tutorials, Vol. 12, No. 4, pp.519-530.

[27] Wei Dong, Chun Chen, Xue Liu, Yunhao Liu, Jiajun Bu, and Kougen Zheng, (2011). SenSpire OS: A Predictable, Flexible, and Efficient Operating System for Wireless Sensor Networks,IEEE transactions on computers, vol. 60, no. 12.

[28] ZHAO Zhi-bin* and GAO Fuxiang, (2009). Study on Preemptive Real-Time Scheduling Strategy for Wireless Sensor Networks, Journal of Information Processing Systems, vol.5, no.3.

[29] Zhou Yu, Jing Bo, (2007). Research and implementation on µC/OS-II operating system into wireless networked sensors, The Eighth International Conference on Electronic Measurements and Instruments, pp. 199-204.