ISSN:2320-0790

An International Journal of Advanced Computer Technology

OS design challenges & research opportunities in real-time WSNs & approach for real time support in Nano-RK

Prof. Manjiri Pathak

Associate Prof., Padm. Vasantdada Patil Pratishthan's College of Engg. Sion, Mumbai, India,
manjiri_pathak@yahoo.com

Abstract: In recent years, wireless sensor network has become an important research domain. WSNs were initially proposed in domains where ordinary networks (not necessarily wired) are not convenient, either because of the missing infrastructures, or when numerous nodes (in the order of hundreds or thousands) are needed to achieve the assigned task. Nowadays WSNs represent a new generation of distributed embedded systems with a broad range of real-time applications. Some of the applications include process control, fire monitoring, border surveillance, medical care, asset tracking, agriculture, highway traffic coordination etc. Such systems need heavy computations & must meet new kinds of timing constraints under severe resource limitations & limited communication capabilities in highly dynamic environments. Bounded end-to- end delay and guaranteed Quality of Service is also expected. So it is highly necessary to have a common software framework that allows smooth and speedy development of the wide range of proposed sensor applications. An operating system can serve this purpose. Operating systems for WSNs should comprise abstractions that handle digital and analog sensors, provide a communication protocol stack, and make efficient use of the system's limited energy capability. Moreover, OSs should provide an interface and a simple configuration system for application developers [2]. The purpose of this survey is to highlight major concerns pertaining to OS design & research challenges in OS for WSNs in real time applications [1].

Keywords: Wireless Sensor Network (WSN), Real-Time Operating System (RTOS), Nano-RK, Wireless Multimedia Sensor Networks (WMSN)

1. Introduction

WSNs challenge many classical approaches to heavy & real-time computing. Wireless networking protocols, operating systems, middleware services, data management, programming models, scheduling, memory management, all fundamentally change when confronted with such new systems and environment. A WSN is a highly dynamic in nature because of node failure due to severe environmental conditions and battery power depletion. WSNs invariably operate in an unattended environment 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, this deviates from traditional OS design. Dense deployment of sensor nodes in the sensing field and distributed processing through multi-hop communication among sensor nodes is required to achieve high quality and fault tolerance in WSNs. Each node is expected to perform a substantial amount of computation related to data filtering, diagnostic, logging, communication, and so on. Many activities, like sampling and actuation, must be triggered periodically and are expected to be executed within bounded delays, otherwise the system failure can be occurred. Typically, a robust Real-Time Operating System (RTOS) providing customizable scheduling policies (for multitasking operation) and reliable services, ranging from networking to peripheral management, is needed in such applications. Because of the severe resource constraints, even simple protocols and algorithms may not perform well when they are actually implemented on the sensor devices; so this challenge is the major factor that is driving current research in WSN OSs. In the next section, we will discuss about various aspects & challenges involved in operating system design for WSN real time applications. In section 3, we will discuss various approaches to address the design challenges in the RTOS like Nano-RK that is used for WSN. In section 4,, we will discuss the research gaps & future scope of research in this field.

2. Issues & challenges:

Traditional OSs are not suitable for WSNs because of many characteristics such as constrained resources, dynamic topology, unattended environment after deployment, diverse data centric applications etc. The operating systems like TinyOS, Contiki, SOS, Mantis OS, LIMOS, Nano-RK, RETOS, LiteOS etc. are especially designed for WSN. From these Nano-RK &

RETOS provide real time support. Now, we will consider some of the issues that should be considered while designing the real time OS for WSN & challenges while fulfilling their requirements.

a) System architecture challenges:

i. Diversity in H/w
ii. Diversity in range of applications
iii. h/w & s/w boundary w.r.t. application[3].
iv. Heterogeneity in the types of sensor nodes

b) High concurrency & limited resources:

As WSNs are event-driven systems, a change in the environment or arrival of a network packet triggers an event-handler on the motes or nodes. As events can arrive while the node is already busy with some data processing, an efficient concurrency mechanism is anticipated to enable sharing of CPU between these two overlapping activities of event handling and data processing. Limited amount of RAM on nodes (typically 4 KB) imposes the real challenge on designing an efficient concurrency model[4]. Multithreading and run-time loading of modules are desirable features of an operating system for sensor network nodes Since each thread has to be allocated with separate stack space, the degree of the concurrent execution of threads is also limited. WSN uses either event driven or multithreaded concurrency model. The multithreaded model involves resource consuming context-switching overhead, and demands locking mechanisms to avoid inconsistency or race conditions. Therefore, to maintain data consistency, the mechanisms are required to ensure the orderly execution of concurrent threads or code that share same data structures or resources. It can be achieved by using the synchronization tools such as semaphores.

c) Use of resources & process scheduling:

Each process uses its own stack space & address space. During context switching & saving the current status of the process while scheduling, more energy is utilized which is not suitable in case of WSN. So the resource allocation such as CPU time and limited memory must be scheduled and allocated properly to the processes to guarantee fairness in their execution.

d) Memory management challenges:

i. 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 [17].

ii. 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 [17]. 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.

iii. **Dynamic memory allocation:** Data memory has been a very constrained resource in sensor networks. 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. Even though the WSN operating systems like MantisOS, SOS and Contiki provide some form of dynamic memory allocation, it becomes important to design efficient dynamic memory allocation techniques for the applications of WSNs moving towards increasing diversity.

iv. Memory protection

The efficient memory protection mechanisms should be provided between different applications of WSNs.

e) Network management challenges & implementation of efficient routing protocols:

i. Network management is the process of managing, monitoring, and controlling the behavior of a network. Wireless sensor networks (WSNs) pose unique challenges for network management. To date, the limited results that have appeared for WSN regarding real-time issues has been in routing. So there is a need for designing routing protocols that guarantee to meet deadlines, support different classes of traffic, deal with the problems such as lost messages, noise and congestion & provide quality of service[10]. In case of WSNs with heterogeneous sensors, complex

routing & communication protocols are required.

ii. Protocols based on conventional TCP/IP stack requires large portions of the RAM, require high bandwidth, and their performance is heavily dependent on the fast processors[5][9]. Because of which, using TCP/IP protocol suite to the resource constrained environment of sensor networks is a challenging task. Many wireless sensor network applications do not work well in isolation; the sensor network must somehow be connected to monitoring and controlling entities which can be done by connecting the sensor network and the controlling entities to a common network infrastructure, so that the sensors and controlling entities can communicate without being physically close to each other & without the need for special proxy servers or protocol converters.

iii. The deployment and mobility of nodes affect the network topology as there is uncertainty on the nodes' location and density. Some of the nodes go down due to severe environmental conditions & battery power depletion. Moreover, some nodes can be added after the deployment of WSN or attached to an object thus making deployment a continuous process. Because of this, the topology also changes many times[11]. So the protocols should be flexible and dynamic in order to react to the different demands of applications. Also the degree and speed of movement can influence in the time the nodes are available and therefore their usability in the network.

f) Dynamic Reprogramming Challenges:

Handling node failures and upgrading software on already deployed nodes in unattended environment & several thousands in number is challenging Therefore, this sensor network must be remotely reprogrammable irrespective of node density and physical accessibility [7]. The feasible solution that allows remote reprogramming must take resource constraints of sensor networks into consideration.

g) Power Management Challenges:

Energy is the most critical resource in WSNs and determines the lifetime of the sensor network[6]. But sensor networks are expected to operate for 3 to 5 years depending on the application for which the nodes are deployed. So the care should be taken to handle this imbalance while designing the OS so that the life time of WSN can be prolonged. The most common way to improve life-time of such network is duty cycling that periodically puts peripherals and CPU into sleep. The challenging factor is to determine the value of the duty cycle and its periodicity as these values. The WSN OS must provide mechanisms to tackle this issue.

h) Keeping separate address space for kernel & the application programs

i) Simulation support:

The simulation support can be provided for testing the performance of a particular application before deployment of WSN for it[6].

j) Design of unified protocol stack:

which is required to improve communication between sensor nodes with different OSs

k) Design of proper database management system for the sensor nodes & efficient query processing

l) Security mechanisms to identify & handle various types of attacks

Because sensor networks may interact with sensitive data and/or operate in hostile unattended environments, making these networks secure is especially challenging because of wireless medium. WSN is mostly unguarded. Hence, capturing a node physically, altering its code and getting private information like cryptographic keys is easily possible for an attacker. Wireless medium is inherently broadcast in nature. Firstly, there are severe constraints on WSNs devices such as minimal energy, limited computational, storage and communicational capabilities, unattended operations etc. Secondly, there is an additional risk of physical attacks such as node capture, tampering & getting private information like cryptographic keys which is easily possible for an attacker[16]. These attacks can disrupt the operation of WSN and can even defeat the purpose of their deployment. Moreover, cryptography based techniques alone are insufficient to secure WSNs. Hence, intrusion detection techniques must be designed and developed to detect any kind of undesirable attacks.

m) Reliability:

In most applications, sensor networks are deployed once and intended to operate unattended for a long period of time. OS reliability is of major concern to proper functioning of WSN [8][9].

In the next session, we will consider how these challenges have been approached in Nano-RK

RTOS, & research gaps in design issues of RTOS.

3. Approach to handle these challenges in Nano-RK RTOS

Nano-RK is known as the energy-aware resource-centric RTOS for sensor nodes. The resources like CPU cycles, sensors, actuators, network buffers, and bandwidth should be used only to the extent that is required by the application. It provides rich functionality and timing support using less than 2KB of RAM and 18KB of ROM. It has a support for multi-hop networking & multitasking [12] & also for extended WSN lifetime.

Now we will see some of the approaches that are considered in Nano-RK to meet the challenges.

a. Use of resources & process scheduling

Nano-RK supports fully preemptive, reservation based fixed priority scheduling[14] with timing primitives to support real-time tasks at two levels: priority scheduling at the process level and priority scheduling at the network level. At any given instance, the highest priority task is scheduled by the operating system[13]. Thus Nano-RK guarantees to meet the deadlines.

b. Memory Management

Nano-RK provides support for static memory management, & not for dynamic memory management. Here, both the OS and applications reside in a single address space.

c. Network management & Communication Protocol Support

For sensor networks the primary goal in network management is minimizing energy use and the main means for doing this is by reducing the amount of communication between nodes, because more energy is utilized for data transfer during the communication between nodes.

Nano-RK supports multi hop networking & provides a lightweight networking protocol stack that provides a communication abstraction similar to sockets. To handle memory more efficiently, transmit and receive buffers are managed by the application. OS copies the received data into the application buffers. Once the data is placed into the application buffer, the application is notified accordingly.

A Time Synchronized Link Protocol, RT-Link provides support for real-time applications through bounded end-to-end delay across multiple hops using Scheduled slots, and collision free transmission [7]. It is implemented over a TDMA link layer protocol, where each node transmits the data in predefined time slots, allowing for energy savings. In case of new mobile node, contention slot is assigned to it using which it makes a reservation request to a gateway. Its membership keeps on changing with time.

d. Resource Sharing

For shared resources such as memory, Nano-RK provides mutexes and semaphores for serialized access. In addition, Nano-RK provides APIs to reserve system resources like CPU cycles, sensors, and network bandwidth, i.e. the tasks can specify their resource needs & OS provides guaranteed access to these resources.[8]

e. Support for Real-time Applications

Nano-RK is a real-time operating system, hence it provides rich support for real-time applications. It supports real-time processes and its offline admission control procedure guarantees to meet deadline associated with each admitted real-time process. Nano-RK provides an implementation of real-time preemptive scheduling algorithms and tasks are scheduled using a rate monotonic scheduling algorithm. Moreover, Nano-RK provides bandwidth reservations for delay-sensitive flows and it claims to provide end-to-end delay guarantees in multi-hop wireless sensor network. Nano-RK is a suitable OS for use in multimedia sensor networks due to its extensive support provided to real-time applications.

f. Power management

Nano-RK supports deep sleep mode by keeping the nodes in a sleep mode when no task to run or to go into a low energy consumption state while still managing its peripherals[15]. It also provides an algorithm like rate harmonized scheduling for energy saving by eliminating CPU idle periods & grouping the execution of different tasks.

In Nano-RK, a resource kernel provides reservations on how often system resources can be allocated. A task might only be allowed to execute with some CPU Reservation, or a node might only be allowed to transmit few packets within a certain amount of time with Network Reservation. It ensures that a node meets its designed battery lifetime as well as protects a failed node from generating excessive network traffic.

g. Watchdog Timer support

Watchdog is a software timer that triggers a system reset action if the system hangs on a crucial faults for an extended period of time. The watchdog mechanism can bring the system back from the nonresponsive state into

normal operation by waiting until the timer goes off and subsequently rebooting the device.

4. Research gaps & opportunities in real time OS design for WSN

In this section, we will consider, 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. As huge amount of data is collected & processed at the nodes, the requirement of secondary memory to store & maintain this data in the database is also increased. To achieve this, much more improvement in file system to manage the secondary storage is required.
- In the existing operating systems for WSN, there is very limited work is done in case of memory management for multiple concurrent applications & support for virtual memory management. Even, RTOS like Nano-RK supports only static memory management. So significant amount of work is required here to support heavy real time data storage & computations. It should also be energy & memory efficient[7]. Especially, in WMSN there is much more work to be carried out in memory management.
- Schedulers have been designed to support soft as well as hard real-time operations in some operating systems, but still much more effort is required in fair scheduling of real time data & to achieve concurrency.
- Still there is a great scope of work in the design of unified networking protocol stack to handle real time traffic with multimedia streams & improve communication between heterogeneous sensors, especially in Wireless Multimedia Sensor network[7].
- APIs for clock synchronization & localization of nodes are required to be provided to optimize system performance. A WSN OS also needs to provide a rich set of basic image and signal processing APIs to support new range of applications with multimedia data.
- Traffic analysis attacks, secure high-level data aggregation, intruder nodes and multiple identity attacks, detection of compromised nodes and privacy concerns constitute some of the most important security challenges [16]. Efficient algorithms are needed to be designed to secure the WSNs against such types of attacks that need to be addressed by WSN in the future.
- More research work is required to design a database management system for sensor nodes & efficient query processing.

5. Summary & Conclusion

OS support is important to facilitate the development and maintenance of WSNs.. 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. So, here various issues related to OS design have to be considered.

In this paper, in the first section, we introduced a role & importance of OS design issues for real time WSN. Then, we considered many issues & research challenges that should be considered while designing the operating system for the same. In the next section, we discussed the approach used to address these challenges in Nano-RK, the Real Time OS used for WSN. Finally, we discussed the research gaps & opportunities for future work in this area.

As the range of possible WSN application domains is growing, lots of improvements & further investigations are required to give stronger real time support & efficient memory management techniques for WSNs. Other issues, as discussed earlier are also under active research & development. Even though the significant amount of work is done in this area, there is still a wide scope for additional work here.

6. References

[1]http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3231431/

[2] http://www.lisha.ufsc.br/Project+CAPES-DFAIT

[3] D. Gay et al., "The nesC language: A Holistic Approach to Networked Embedded Systems," In Proc. of ACM SIGPLAN Conference on Programming Language Design and Implementation, 2003

[4] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt, "Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors," In Proc. of First IEEE Workshop on Embedded Networked Sensors, November 2004.

[5]http://www.cs.berkeley.edu/_kwright/nestpapers/ammote.pdf.

[6] S. Bhatti et al., "MANTIS OS: An Embedded Multithreaded Operating System forWireless Micro Sensor Platforms," ACM/Kluwer Mobile Networks and Applications (MONET), Special Issue on Wireless Sensor Networks, vol. 10, no. 4, August 2005.

[7] http://www.mdpi.com/1424-8220/11/6/5900

[8]http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5462978&url=http%3A%2F%2Fieeexplore.ieee.org

[9]http://tif.bakrie.ac.id/pub/proc/icacsis2011/pdf/301.pdf

[10]http://www.rimtengg.com/iscet/proceedings/pdfs/misc/176.pdf

[11]http://paginas.fe.up.pt/~prodei/DSIE08/papers/41.pdf

[12] http://www.nanork.org/projects/nanork/wiki
[13] http://en.wikipedia.org/wiki/Nano-RK
[14]http://moss.csc.ncsu.edu/~mueller/rt/rt11/readings/projects/g4/finalreport.pdf
[15]https://www.msu.edu/~liyang5/docs/nanork_tutorial.pdf
[16] Manjiri Pathak. 'Issues in Security & Intrusion Detection in Wireless Sensor Networking', International Conference on Advanced Technologies for Research & Product Development (ICATRPD 2012)
[17] http://www.comp.nus.edu.sg/~doddaven/cata.pdf