ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology
Volume 1, Issue 4, June 2012

Current Scenario in WSNs

Vandana Jindal   Anil Kumar Verma   Seema Bawa
Thapar Univ., patiala

Abstract

WSNs have a number of sensor nodes which are extremely small in size, consuming low power and costing less. WSNs overweigh traditional network in the areas of deployment, scalability, ease of use and mobility. Due to lack of structure and resources, WSNs have a few limitations as compared to the traditional network. Middleware is a software that fills the gap or glues together the network hardware, network stacks, Operating System and applications. A middleware should facilitate development, deployment, execution and maintenance of sensing based applications. It should provide ways to achieve maximum utilization of system resources; an environment that is capable of handling various applications. A middleware is an approach that meets the design and implementation challenges of WSN technologies. Each device (i.e., sensor node) requires an operating system (OS) which will act as a manager in controlling the sensor nodes helps in filling the gaps between application and the hardware and last but not the least will provide hardware abstraction to the application software. These Oss are totally unlike our traditional OSs. OS a resource manager comprising of various programs that help in managing various resources like memory, I/O devices, file system etc. is not at all suitable for WSNs. This is so because in WSNs the sensors are constrained with limited resources like memory, power, dynamic topology and various data centric applications.

Key Words – WSN,Mate, Impala, Tinydb, SINA, DsWare, MILAN, Mire, IrisNet, DFuse, Motes, I Motes, Spec, Stargate, EMERALDS, EYES OS, Magnet OS, MANTIS, OSPM, PicOS, TinyOS, Contiki.

1. Introduction

A Wireless Sensor Network (WSN) [1] is a wireless network having distributed self-organized autonomous devices employing sensors to monitor environmental conditions like temperature, humidity, pressure, movement etc. Dense deployment of disposable and low-cost sensor nodes makes WSN concept beneficial for battle-fields. Tiny sizes and light weight structure of these (WSN) nodes provides many functionality in health applications like drug administration, tracking and monitoring doctors and patients as well as commercial applications like home automation for smart home environments, detecting and monitoring burglary, vehicle tracking and detection etc. WSN is a means to bridge a gap between the physical and the virtual world. The working of WSNs is very different from traditional computer networks, due to their tight integration with the physical world. This paper summarizes our initial investigation with regard to middleware and operating system used in WSN. By no means do we claim that the search has been exhaustive but have attempted to cover maximum areas and parameters.

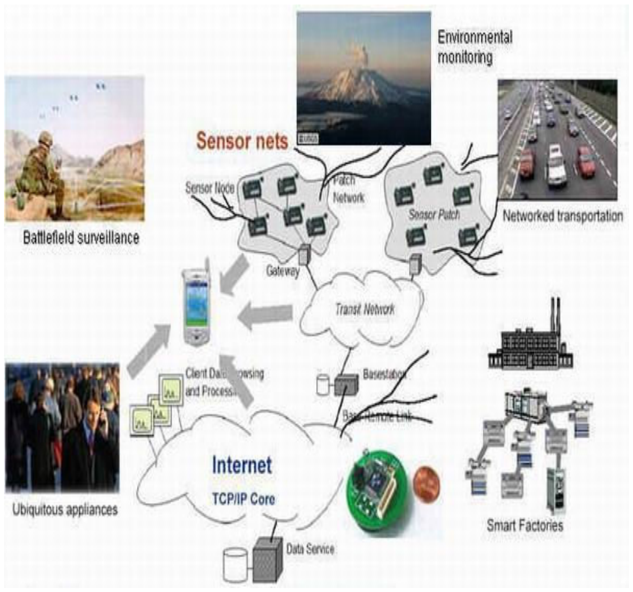

Fig. 1: A Wireless Sensor network [19]

2. Wireless Sensor Networks (WSNs)

WSNs have a number of sensor nodes which are extremely small in size, consuming low power and costing less. WSNs overweigh traditional network in the areas of deployment, scalability, ease of use and mobility [2]. Due to lack of structure and resources, WSNs have a few limitations as compared to the traditional network. To name a few – constraints in terms of energy, processing power, memory, complicated structure topology etc.
However, it is not a small task to develop applications for WSNs due to various impediments of sensor nodes. A middleware layer is a small approach to fully meet the challenges of WSN. Middleware is a link between application and low level operating system helps in solving many WSN issues and enhanced application development [3]. It also acts as layer between user type server and the sensor nodes. The main purpose of the middleware is to support the development, maintenance, deployment and execution of sensing-based applications.

3. Middleware for WSNs

3.1 Need for Middleware

Middleware is software that fills the gap or glues together the network hardware, network stacks, Operating System and applications. A middleware should facilitate development, deployment, execution and maintenance of sensing based applications. It should provide ways to achieve maximum utilization of system resources; an environment that is capable of handling various applications. Middleware is an approach that meets the design and implementation challenges of WSN technologies.

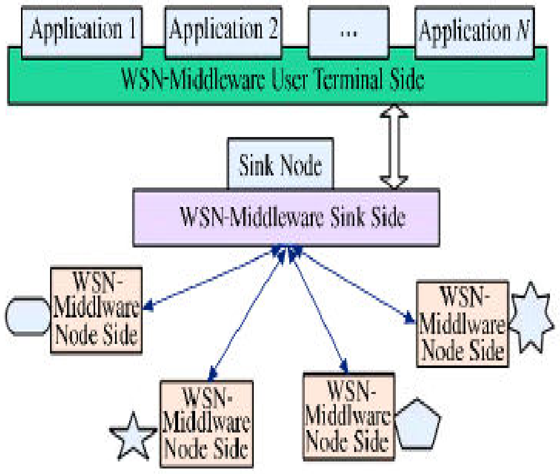

Fig.2 Middleware for WSN

3.2 An Overview of existing Middleware approaches

The middleware approaches may be broadly categorized into seven classes based upon their architecture and approach mechanism employed.
Programming sensor network is classified into 2 categories:

1. **Programming support**: it provides system, services and runtime mechanism (code distribution, execution etc.).
2. **Program abstractions**: it provides abstractions of sensor networks and sensor data.

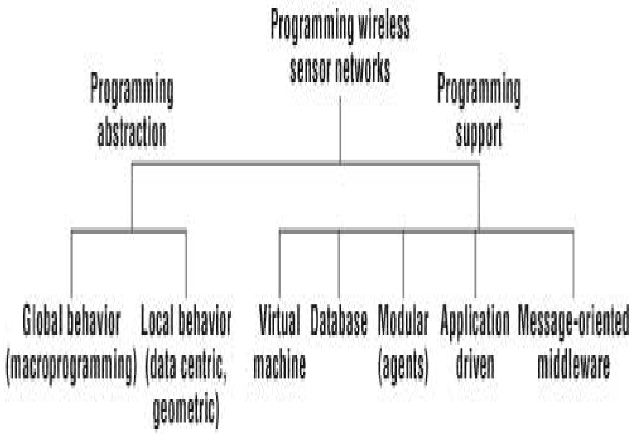
Programming wireless sensor networks
Programming abstraction
Programming support
Global behavior (macroprogramming)
Local behavior (data centric, geometric)
Virtual machine
Database
Modular (agents)
Application driven
Message-oriented middleware

Fig. 3 Taxonomy of programming models for WSNs.

- **Application Driven:**

This approach adjusts the network configuration according to the requirements of the application. The structure provides various network configurations by adopting suitable protocol. Combinations of sensors and network configurations meet the application requirements exhibiting varied performance of QoS. E.g. MiLAN(Middleware Linking Application and Networks). It provides the solution that particular applications are permitted to affect the entire network‟ s performance. MiLAN is originally designed for medical advising and monitoring [4].

- **Component based:**

Binding the components forms the basis of this approach. The cross-platform components can be employed in various platform environments by making use of middleware API. This approach facilitates running of applications on different devices. E.g. Runes – Reconfigurable Ubiquitous Networked Embedded Systems. It can be deployed on devices with different platform, from PCs to sensor node with ContikiOS [5].

- **Distributed Database:**

This approach treats the whole sensor network as a distributed database. It has a user friendly interface, employing SQL. It is good at regular querying but does not support real-time applications. E.g. TinyDB [6], Cougar [7], DsWare (Data Service Middleware) [8]. Cougar applies database pattern in sensor network.

- **Macroprogramming:**

This approach permits the programmer to program the entire sensor network, making high-level specification and dealing with low-level concerns at each node in the sensor network [9]. E.g. COSMOS - It is a novel architecture for macroprogramming heterogeneous sensor network systems involving aggregate system behavior specification rather than device-specific applications [10].

- **Message Oriented:**

This approach implements the communications using publish/subscribe mechanism between node applications to user side. With the help of publish/subscribe service (in the middleware) the messages are exchanged between the message sender and the related receiver. E.g. Mires- The key components is the publish/subscribe service acting as a middleware to transfer publish/subscribe message and establishing communications between local nodes to user application [11].

- **Mobile Agents:**

The main feature of this approach is that the applications are treated as modules for injection and through the network using mobile codes. It proves to be very efficient in dynamic applications, consuming very less amount of power. E.g. Impala- a middleware that makes use of modular programming approach. It finds its application in (ZebraNet Project) wild-life tracking where communication is not easy [12].

- **Virtual Machine:**

This approach makes use of virtual machines and interpreters. The applications are written into small modules. These modules are then injected and distributed through network, finally finding their way to the virtual machine, where these are interpreted, for implementing the application. E.g. Mate- It runs on the top of TinyOS operating system, and works as a byte code interpreter. It has a high level user interface, and breaks large programs into small pieces called capsules. Injection of program into sensor network is faster and done with ease, with the help of **Mate** [13].

3.3 Desirable Characteristics in a middleware

The evaluation of any middleware may be done, based on the following desirable characteristics i.e. usability, mobility, scalability, power saving and heterogeneity.

- **Usability:** The middleware should be easy to use. Middleware having high-level and implement GUI are strong support on usability where as middleware employing instruction set and deep understanding on structure are weak support on usability. Database middleware using high-level language and SQL like interfaces are easy to use.
- **Mobility:** Connectivity between the mobile nodes is the most important factor, as it finds its application in tracking problems. The middleware should possess the ability to keep communications with mobile sensor nodes in WSNs. Middleware having a set of efficient ad hoc routing protocols are strong support on mobility.
- **Scalability:** If an application grows, the network should be flexible enough to allow this growth anywhere and anytime without affecting network performance. Maintenance and reconfiguration of sensor nodes is a must in dynamic topology in a middleware.
- **Power Saving:** Sensor nodes which are fast approaching a cubic millimeter in volume, can store or harvest limited amount of energy from the environment. Middleware should have mechanisms or techniques to manage electrical devices and reduce the power consumption. Data transmission overhead plays a vital role for power saving. Less the data transmission, less is the energy consumption.
- **Heterogeneity:** A middleware should possess techniques to implement cross-platform applications enabling them to be used for a wide range of hardware and devices.

Based on the various approaches and desirable characteristics of the middleware, many have come up in the market for catering to our needs.

4. Challenges and Design principles

i) Real World Integration: Usually the applications are involved with real-time phenomena. The middleware should be such that it should be capable enough to impart real-time services.
ii) Application Knowledge: In order to get the results for a particular application, mapping has to be performed between the application and the network. The middleware should be designed in such a way that it maintains equilibrium between these applications specificity and middleware generality.
iii) Limited Power: The sensors do not have abundance of resources like the memory, battery life etc. In order to make optimum use of resources, the size of the deployed OS should be less, it should enter into the sleep mode when not performing any executions as the battery life is very less.
iv) Scalability
v) Mobility
vi) Heterogeneity
vii) Dynamic network organization: Middleware should be capable of supporting adhocism [14] i.e., detecting resource and its location.
viii) Data aggregation: We know that as thousands of sensors are deployed within a small range of area for sensing the changes in the data, it results in a large amount of redundant data which in turn leads to high communication costs (because energy consumed in transmitting a single bit is same as executing 1000 of instructions). Middleware should be capable of aggregating the data [15], thus eliminating the redundancy and minimizing the number of transmissions.
ix) Security: Efforts should be made for inducing security during the initial design phase of the software.
x) Quality of service (QoS): It may be viewed as [16]

- Application specific i.e., it includes sensor node measurement, deployment coverage etc.
- Network i.e., how network can meet the application needs by making efficient use of network resources like bandwidth, energy.

5. Metrics involved

Several performance metrics to evaluate middleware includes:

- **Response time CPU**
- **Utilization**.
- **Reprogramming**
- **Processing cost**.
- **Scalability Performance**
- **Mobility**
- **Heterogeneity**
- **Usability**

6. Types of Sensors

The sensor nodes may be divided into four groups:

1. **Motes** [17] - These fall under the category of **generic sensing platform**.
2. **I Motes** [18] – These are **high bandwidth sensing platforms** which handle sensed data flows.
3. **Spec** [18] – These are **specialized sensing platforms** having a single chip.
4. **Stargate** [19] – These fall under the category of **Gateway platforms**. The node is used as a sink which can further connect low-level sensor nodes to the internet.

Differences amongst all the above mentioned devices lie with respect to:

- Function of the sensor
- Frequency of the microprocessor
- Memory size
- Transceiver bandwidth
- Characteristics

Similarities amongst these devices lie with respect to

- Hardware components

i.e., physical sensor, microprocessor or microcontroller, memory, a radio transceiver and a battery. All the above mentioned components should be arranged accordingly so that these operate effectively and correctly.
Each device (i.e., sensor node) requires an operating system (OS) which will act as a manager in controlling the sensor nodes helps in filling the gaps between application and the hardware and last but not the least will provide hardware abstraction to the application software. These OSs are totally unlike our traditional OSs.

Based on the various constraints and characteristics of the sensor nodes, many OSs have come up in the market for catering to our needs. From amongst those available we shall evaluate a few.

7. OS Design Issues

OS a resource manager comprising of various programs that help in managing various resources like memory, I/O devices, file system etc. is not at all suitable for WSNs. This is so because in WSNs the sensors are constrained with limited resources like memory, power, dynamic topology and various data centric applications. Hence, a new type of OS is required with regard to these special

1. **Memory management**: In contrast to traditional OS where memory is assigned to a particular task, in WSN the size of the memory is very small.
2. **Process Management**: Context switching and data copying are the highlighted feature of traditional OS which undoubtedly is the most unsuitable foe WSN as they pose to be energy inefficient for them.
3. **Application Program Interface (API)**: The nodes in WSNs for executing an operation for an application must be enabled with modular and general APIs.
4. **Kernel Model**: The WSNs use event driven and Finite State Machine (FSM) models for designing microkernel for WSNs.
5. **No External Disk**: Since no external disk is available, the OS for WSNs cannot have a file system.
6. **Code programming**: The sensor nodes and their algorithms need to be adjusted for their functionality or for energy conservation so the OS should be able to reprogram and upgrade them.

8. Desirable characteristics of Sensor OS (SOS)

Considering the resource constraints of the sensor nodes the most desirable features an OS should possess, are as follows:

1. SIZE should be small as nodes have just 10-100 kilobytes of memory.
2. Efficient resource management should be provided.
3. Should provide real-time support, as there are real-time applications.
4. For controlling the hardware, it should support generic programming interface.
5. As the functions performed by the sensor nodes may need to be changed so the OS should provide reliable and efficient code distribution.

It should be able to provide power management thus enhancing the performance and lifetime of the system.

9. Expected features of the next generation WSN OSs

1. Power aware policies
2. Self organization
3. Easy interface to expose data
4. Simple way to program, update and debug network applications

10. Conclusion

Various frameworks have been evaluated based on the following criteria: heterogeneity, scalability, power awareness, mobility, ease of use, and openness. The design of a middleware layer for sensor networks that fully meets the challenges is still an open issue to discussion. The various operating systems mentioned above are able to overcome not all but a few constraints. Following guidelines may be concluded:

- Operating system should be portable.
- The operating system should provide a way for testing and debugging application programs.
- Because of the constrained resources in embedded systems, main idea is to build an, efficient code, leading to low power consumption.
- Designing a small code so that less of memory may be utilized.
- Self organization and reorganization should be the motive of the operating system during node failure.
- When large number of nodes are deployed their behavior must be like that of distributed databases.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E.Cayirci, “Wireless sensor networks: A survey,” *Computer Networks*, vol. 38, pp. 393–422, 2002.

[2] Wikipedia.en.wikipedia.org/wiki/wsn

[3] S.Hadim and N.Mohamed, "Middleware challenges and approaches for Wireless sensor networks", 7th March 2006.

[4] A.Murphy and W.Heinzelman. "Milan: Middleware linking Applications and networks", Technical report, 2002.

[5] P. Costa, G.Coulson,C.Mascolo, G.P.Picco, and S.Zachariadis, "The runes middleware: A reconfigurable component-based approach to networked embedded systems", technical report, 2005.

[6] S.R.madden, M.J.Franklin, J.M.Hellerstein, and W.Hong. Tinydb: "An aquisitional query processing system for sensor networks", June 2005.

[7] P.Bonnet, J.Gehrke, and P.Seshadri, " Towards sensor database systems", *In Proceedings of the Second International Conference in Mobile Data Management.*

[8] S. Li, Y. Lin, S. H. Son, J. A. Stankovic, and Y. Wei. "Event detection services using data service middleware in distributed sensor networks", In Telecommunication Systems, volume 26, pages 351–368, 2004.

[9] S.Hadim and N.Mohamed. "Middleware for wireless sensor networks: A survey", *In first International Conference on Communication System software and middleware.* IEEE Computer

Society, 2006.

[10] Meyer, B, " Applying design by contract", *IEEE Computer* 25, October 1992, pages 40-51.

[11] E.Souto, G.Guimaraes, G.Vascoucelos, M.Vierra, N.Rose, C.Ferroz, and J.Kelner. "Mires: a publish/subscribe middleware for Sensor networks", October 2005.

[12] T.Liu and M.Martonosi. "Impala: A middleware system for managing autonomic, parallel sensor systems", *In Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming*. ACM Press.

[13] P. Levis and D. Culler, "Maté: a tiny virtual machine for sensor networks," in *Proceedings of ASPLOS*, 2002.

[14] Q. Jiang and D. Manivannan , "Routing Protocols for Sensor Networks,"*Proc. 1st IEEE Consumer Comm. and Networking Conf.* (CCNC 04), IEEE Press, 2004,pp. 93-98.

[15] B. Krishnamachari , D. Estrin and S. Wicker , "Impact of Data Aggregation in Wireless Sensor Networks,"http://doi.ieeecomputersociety.org/10.1109/ICDCSW.2002.1030829, *Proc. 22nd Int'l Conf. Distributed Computing Systems* (ICDCSW 02), IEEE CS Press, 2002, pp. 575-578.

[16] D. Chen and P.K. Varshney , "QoS Support in Wireless Sensor Networks: A Survey,"*Proc.Int'l Conf. Wireless Networks* (ICWN 04), CSREA Press, 2004, pp. 227-233.

[17] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister,
„„*System Architecture Directions for Networked Sensors*,‟ ‟ ACM
SIGOPS Operating Systems Review, Vol. 34,No. 5, December 2000, pp. 93–104.

[18] J. Hill, M. Horton, R. Kling, L. Krishnamurthy, „„The Platforms Enabling Wireless Sensor Networks,‟ ‟ Communications of the ACM, Vol. 47, No. 6, June. 2004, pp. 41–46.

[19] http://en.wikipedia.org/wiki/Wireless_Sensor_Networks

Vitae

Vandana Jindal is currently working as an Assistant Professor in the department of Computer Science at D.A.V College, Bathinda. She holds degrees of B.Tech, MCA, M.Phil. Since January 2009, she has been with the Thapar University, Patiala in Punjab as a Ph.D. student. Her research interests include database management systems, wireless sensor networks. She is a member of IEEE and IEI.

A K Verma is currently working as Assistant Professor in the department of Computer Science and Engineering at Thapar University, Patiala in Punjab (INDIA). He received his B.S. and M.S. in 1991 and 2001 respectively, majoring in Computer Science and Engineering. He has worked as Lecturer at M.M.M. Engg. College,Gorakhpur from 1991 to 1996. From 1996 he is associated with the Thapar University. He has been a visiting faculty to many institutions. He has published over 100 papers in referred journals and conferences (India and Abroad). He is member of various program committees for different International/National Conferences and is on the review board of various journals. He is a senior member (ACM), MIEEE, LMCSI (Mumbai), GMAIMA (New Delhi). He is a certified software quality auditor by MoCIT, Govt. of India. His main areas of interests are: Bioinformatics, database management systems and Computer Networks. His research interests include wireless networks, routing algorithms, securing ad hoc networks and design/develop applications for other fields based on IT.

Seema Bawa holds M.Tech (Computer Science) degree from IIT Kharagpur and Ph.D. from Thapar Institute of Engineering & Technology, Patiala. She is currently Professor and Dean of Students' affairs (DOSA). Her areas of interest include Parallel and distributed computing, Grid computing, VLSI Testing and network management. Prof. Bawa is member of IEEE, ACM, Computer society of India and VLSI Society of India.