Contemporary Engineering Sciences, Vol. 7, 2014, no. 13, 649 - 660
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ces.2014.4671

Middleware Systems for Wireless Sensor Networks: A Comparative Survey

Xiang Li and Sangman Moh

Dept. of Computer Eng., Chosun University
Gwangju, South Korea

Abstract

Wireless sensor networks (WSNs) play an important role of the Internet of things as one of the core technologies in ubiquitous and pervasive computing. Middleware for WSNs is a hot topic as applications and platforms in WSNs have been widely developed. This paper comparatively reviews middleware systems for WSNs. In particular, the advanced middleware systems reported recently are reviewed and all the existing middleware systems are compared in terms of important features and characteristics. First, challenging issues about the important features for designing middleware for WSNs are introduced. Then, recent advancements in middleware for WSNs are reviewed and compared.

Keywords: Wireless sensor network, middleware, data fusion, database, adaptability

1. Introduction

A wireless sensor network (WSN) consists of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure, etc., and to cooperatively deliver their data through the network to the sink node. Initially, the development of wireless sensor networks was motivated by military applications like battlefield surveillance. Today such networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on.

In WSNs, processing power, storage, and communication capabilities of sensor nodes are relatively weak aspects, and sensor nodes are powered by small batteries. The three functions of data collection, processing and transmission need to be well balanced and cooperatively carried out by every sensor node. As a bridge between operating system and applications, middleware in WSNs plays important roles in data fusion, resource management, security, quality of service (QoS) support, etc [1-4]. Even in middleware, energy conservation is one of the most important factors in battery-operated WSNs.

In this paper, design issues and recent advancements in middleware for WSNs are extensively surveyed and discussed. The advanced middleware systems reported recently are reviewed and all the existing middleware systems are compared in terms of important features and characteristics. First, important features and design issues in middleware for WSNs are addressed in brief. Then, the various approaches and systems of middleware for WSNs are reviewed and compared with each other.

The rest of this paper is organized as follows. In the next section, the key issues in the design of middleware for WSNs are introduced. In Section 3, middleware systems, including recent advancements, are reviewed and compared. Finally, this paper is concluded in Section 4.

2. Challenging Issues

In this section, important design issues in middleware systems for WSNs are summarized in brief.

2.1. Data Fusion

Most applications in WSNs need a large number of sensor nodes to effectively collaborate on information gathering, target surveillance and sensing the environment. Accordingly, in the process of collecting information, it is inappropriate for each node to transmit sensed data to the sink directly. In WSNs, the sensed information is inherently redundant, and such redundancy wastes a lot of bandwidth and precious energy. Data fusion refers to the process whereby multiple copies of data or information are combined to more efficiently serve the needs of the end users. In most WSN applications, under normal circumstances, monitoring the results does not require a lot of redundant raw data, and thus, data fusion is an effective means to deal with the problem.

2.2. Resource Management

The nodes in WSNs are usually powered by batteries, but the batteries are not easy to replace. So, how to effectively use the limited energy is one of the most concerning problems. In order to maximize the lifetime of WSNs, a variety of algorithms and protocols must be taken into account so they can use the limited energy effectively. Once a system is designed, energy use can be reduced further by

dynamic power management [2-3]. Dynamic power management is sometimes called dynamic energy management. In addition, dynamic voltage scaling is another approach to dynamic energy saving.

2.3. Scalability and Network Topology

In WSNs, because of the increased number of nodes and dynamic node mobility, node failure and communication failure will readily affect network topology and scalability. The middleware should be flexible enough to deal with these kinds of network issues [3-4].

2.4. Heterogeneity

Middleware should provide low-level programming models to meet the major challenge of bridging the gap between hardware technology and the broadly needed activities.

2.5. Application Knowledge

In the design of middleware, a wide range of applications and the problem of limited resources should be taken into account. It is not simple to take information about applications into middleware design. Even so, middleware is designed to support a wide range of applications [2-4].

2.6. Security

WSNs are commonly used in military reconnaissance, medical and rescue forecast systems, etc. In many cases, WSNs are vulnerable to various security threats, such as eavesdropping, cajoling, imitation, injection. replay, denial of service, etc.

2.7. Quality of Service (QoS)

A WSN is composed of a large number of sensor nodes to accomplish a specific function. It uses self-organizing multi-hop communication from sensors to the sink. QoS metrics are bandwidth, loss rate, throughput, etc. Middleware needs to take into account the bandwidth allocation, availability, and so on, in order to ensure reliable and qualified service.

2.8. Limited Power

More than half of power consumption in a node is spent on wireless communications. The wireless communication comprises four operation modes: transmitting mode, receiving mode, idle mode and sleep mode. During sleep mode, no communication is enabled, and thus, little energy is consumed. However, it is limited to reduce wireless communications.

3. Middleware Systems for WSNs

3.1. Programming Models

Middleware is a very broad category, addressing different application requirements. Because middleware needs to support heterogeneous and distributed environments of both operating systems and network protocols, it must be able to provide a distributed environment for communications services. Middleware can be categorized into different groups in accordance with different programming models in WSNs. That is, various middle systems are classified into five different programming models of virtual machine, database, modular, application-driven and message-oriented approaches. The programming models in WSNs and the corresponding middleware systems are shown in Fig. 1.

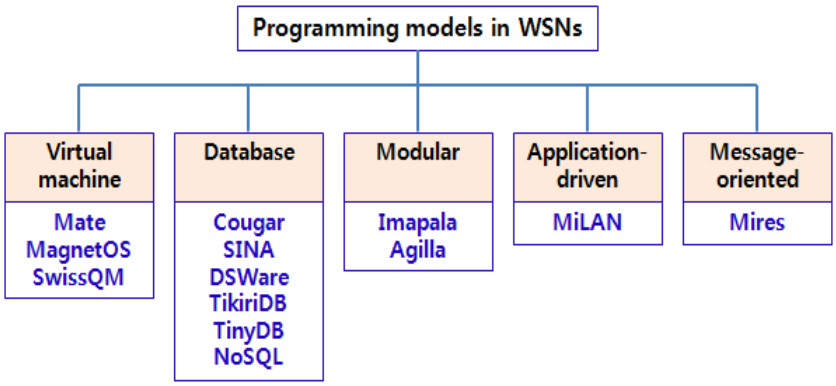

Fig. 1. Programming models and middleware systems in WSNs.

3.2. Middleware Systems

3.2.1. Virtual Machine Approach

Generally, through the virtual machine, the interpreter and the agent, this approach provides a virtual machine environment in order to simplify application development and deployment.

A. Mate

Mate [5] runs on the TinyOS virtual machine. Mainly focused on solving or overcoming bandwidth and energy constraints of WSNs, Mate effectively handles dynamic change in the network environment to provide better interaction and adaptability. Through the virtual machine, Mate provides users with an interface that supports heterogeneous networks and sensor data access functions. Mate uses synchronous mode to support packet transmission and timeouts, etc. Synchronous mode makes application programming simple.

B. MagnetOS

MagnetOS [6] is a distributed, adaptive, power-aware operating system for ad hoc and sensor networks. It also includes functions of middleware. It is a unifying, stand-alone system for ad hoc and sensor networks allowing them to be adaptive, efficient, general-purpose, and scalable.

The system adapts to resource constraints and changes in the underlying network in a rapid, yet temporally stable, manner. In particular, it adapts to changes in network topology, application behavior, and available resources. Policies and mechanisms used for adaptation yield good power utilization and increase system longevity, without excessive communication or power consumption.

The system executes applications over networks of nodes with heterogeneous resources and capabilities, as well as over both ad hoc and fixed nodes. Changes in underlying hardware or software choices as well as network infrastructure and physical characteristics do not require programmer involvement. The system and its underlying algorithms for optimization and adaptation work both on a single node and across a large network.

C. Scalable Wireless Sensor Network Query Machine

SwissQM [7] is a stack-based virtual machine for WSNs. It was created at the Department of Computer Science at ETH (Eidgenössische Technische Hochschule) Zürich. SwissQM defines a bytecode instruction set that is independent of sensor platforms and application languages. Traditionally, WSNs are programmed using low-level languages, such as nesC, on systems such as TinyOS. This is, in general, difficult because the programmer needs to take care of many low-level details, which vary between different platforms. Creating bytecode rather than nesC code increases the abstraction level of programming sensor networks. In combination with a gateway system that accepts programs and queries written in a high-level language and translates them into bytecode, programming of WSNs can further be simplified. SwissQM is a flexible platform for programming WSNs and data acquisition tasks. SwissQM allows event processing at the sensor nodes; implementation of data-cleaning pipelines; finite state automata at the sensor nodes, pushing down complex processing functionality all the way to the sensor nodes; user-defined functions; a turing-complete programming model; and programming sensor networks using different high-level language front ends.

3.2.2. Database Approach

A. Cougar

Cougar [8, 9] is perception data query middleware as a virtual relational database. Cougar uses a Structured Query Language (SQL)–like language to check sensor network management operations. System data are stored as the characteristics of its contents. In addition, Cougar's virtual abstract data type functions with associated attributes. Cougar is composed of a three-tier architecture, query agent, front-end components, and a graphical user interface. In addition, Cougar supports in-network processing, reduces energy consumption, and improves

the life cycle of WSNs. Cougar is particularly suitable for large-scale sensor networks.

B. Sensor Information Networking Architecture (SINA)

SINA [10] is based on spreadsheet database querying and monitoring. The spreadsheet method is abstract, and allows information management in order to meet the changes and demands of the application. In SINA, hierarchical clustering, an attribute-based naming mechanism and a data-centric operating mechanism are used. Each data element represents a sensor node attribute. The data element is unique, each sensor node maintains a data table, and the entire sensor network can be seen as a series of data table polymerizations. The query language can directly manipulate the underlying sensing hardware, includes location identification, and offers a complete communication function.

C. Data Service Middleware (DSWare)

DSWare [11] provides data-centric and group-based services for WSNs. The real-time event service handles unreliability in individual sensor reports, correlation among different sensor observations, and inherent real-time characteristics of events. The event service supports confidence functions that are designed based on data semantics, including the relative importance of sub-events and historical patterns. When the failure rate is high, the event service enables partial detection of critical events to be reported in a timely manner. It can also be applied to differentiate between the occurrences of events and false alarms. It supports packet-based decision-making and reliable data storage, making it adaptable. It also supports data storage, caching, packet management, event detection, data ordering and scheduling operations.

D. TikiriDB

TikiriDB [12, 13] is a database abstraction layer for the Contiki operating system. TikiriDB provides functionalities for retrieving data from Contiki OS–based WSNs without needing programming knowledge of sensor nodes. TikiriDB provides a MySQL-like query interface. Clients can issue queries and retrieve corresponding results with TikiriDB. Clients provide a graphical view of queries and query results (using graphs and charts).

The TikiriSQL library gets client-issued queries, parses those queries, and generates query packets. The TikiriDB library gets result packets from Serial-Forwarder and sends them to the client. Serial-Forwarder listens to query packets sent by clients and sends them to a sensor mote that connect to a USB port. It uses a linked list to manage multiple client requests. The mote connected to the USB port reads query packets sent from Serial-Forwarder.

E. TinyDB

TinyDB [14, 15] is the most primitive query processing database under the TinyOS operating system and was developed at the University of California,

Berkeley. It is a distributed query processing method to handle queries with energy efficiency, using a sensor-based query language similar to SQL. In TinyDB, the routing tree is configured among the sensor nodes using a range of sensor nodes and sensing information as well as by using energy and reducing power consumption. TinyDB does not depend on the nesC language. A SQL interface is used to extract data of interest from the sensor nodes. The residual energy of sensors can be also retrieved effectively in TinyDB.

F. Not Only SQL (NoSQL)

The concept of NoSQL [16, 17] started in 1998. NoSQL databases are different from relational database management systems (RDBMS). An RDBMS is designed to guarantee atomicity, consistency, isolation, durability (ACID) properties. But NoSQL databases do not guarantee ACID properties. They are designed basically for performance and scalability. Normally, NoSQL databases are suitable for large sets of data. Working with a large set of data using a table-based database system, it needs a lot of resources to store such massive data, and operations are time-consuming. With NoSQL databases, handling a massive amount of data is much easier, and performance is better compared to an RDBMS. The only limitations with NoSQL are memory and processing speed. NoSQL database systems use a key and value pair to store data. So, if one wants to keep data in a persistent state and have access to them, this would be an ideal database system.

3.2.3. Modular Approach

A. Impala

Impala [18] is intended to act as an operating system, resource manager and event filter, on top of which specific applications can be installed and run. Its major contributions are (i) to explore the implementation of a non-VM-based middleware layer intended for infrequent code updates and (ii) to link the functionality for application adaptability and application updates, offering a framework that works well for both. In addition to simulating large-scale sensor deployments, a full system was prototyped and evaluated [18].

B. Agilla

Agilla [19] is the first mobile agent middleware for WSNs implemented entirely on TinyOS. Agilla is mobile agent–based middleware with a stack-based architecture. Its stack-based architecture reduces code size. Agilla allows agents to move from one node to another using two instructions: clone and move. Up to four agents can run on a single sensor node. Since one node can run multiple agents at the same time, multiple applications can be simultaneously supported on the network. To save energy, Agilla can move its agent to bring computation closer to the data rather than transmitting the data over an unreliable wireless network. Agilla does not have a policy for authenticating or monitoring agent activities. Also, its assembly-like and stack-based programming model makes programs difficult to read and maintain.

3.2.4. Application-Driven Approach

A. Middleware Linking Application and Networks

MiLAN [20] is focused on high-level concerns by providing an interface that is mainly characterized by applications that actively affect the entire network. MiLAN lets sensor network applications specify their quality needs, and it adjusts network characteristics to increase application lifetime while still meeting those needs. To accomplish that, it receives the individual application's QoS requirements over time and provides a way to meet these QoS requirements using different sensor combinations. It changes over time, receiving application QoS requests, to identify from the network how to meet the QoS level of the sensor set. MiLAN has self-organizing mechanisms working on a real-time basis to reflect dynamic changes in the network. It also takes into account the active needs of the application.

3.2.5. Message-Oriented Approach

Message-oriented middleware (MOM) is an infrastructure that uses a common communication channel in the transmission of data between applications. In a communication environment based on MOM, messages can be sent and received asynchronously. And the messaging system forwards messages to the final destination.

A. Mires

Mires [21] successfully demonstrates that conventional, easy-to-use message-oriented middleware is applicable in WSNs. Its asynchronous communication model uses a publish–subscribe mechanism that confers considerable energy savings. Because Mires is built on TinyOS, it supports hardware heterogeneity and openness. TinyOS's event-based and message-oriented nature makes it a suitable foundation on which to build publish–subscribe middleware. Mires addresses scalability and mobility only partially, since the multi-hop routing protocol that Mires uses does not include a resource-discovery mechanism. Furthermore, its impact on the network's overall performance must be evaluated.

4. Comparison and Discussion

Table 1 compares the different aspects of the existing middleware systems. The attributes include abstraction, data fusion, resource management, dynamic topology, application of knowledge, security, QoS, scalability, mobility, heterogeneity, and usability. These 14 attributes represent the features and characteristics of middleware systems in WSNs. Virtual machine–based middleware runs on each node, resulting in more energy overhead. Database-based middleware is relatively heavy.

Table 2 summarizes and compares the statistics collected from Table 1. Each cell in Table 2 shows the number of attributes in the category. From Table 2, the six middleware systems of Mate, MagnetOS, SwissQM, Impala, Agilla and Milan are superior to the others in terms of the number of fully supported attributes. In other words, the virtual machine–based approach, modular programming approach, and application-driven approach are preferable. Even so, choosing a middleware system for a WSN is largely dependent upon what application is run on the WSN.

5. Conclusions

In this paper, the design issues and recent advancements of middleware systems in WSNs have been revisited, categorized, and compared. According to our qualitative comparison, the six middleware systems of Mate, MagnetOS, SwissQM, Impala, Agilla and MiLAN are superior to the others in terms of the number of fully supported attributes. However, choosing a middleware system for a WSN is dependent on its main application.

Even though many middleware systems have been researched and developed so far, there exist some problems, such as lack of flexibility to support different protocol stacks and QoS. The middleware in WSNs is still a hot topic, and there is a lot of research still to be done. Middleware development will help promote the development and deployment of WSNs.

Table 1. Comparison of middleware systems for WSNs.

Middleware system	**Abstraction**	**Data fusion**	**Resource management**	**Dynamic topology**	**Application knowledge**	**Programming paradigm**	**Adaptability**
Virtual machine–based approach							
Mate	Y	N	Y	Y	N	Y	N
MagnetOS	Y	Y	Y	Y	Y	N	Y
SwissQM	Y	Y	Y	N	N	Y	Y
Database-based approach							
Cougar	Y	Y	Y	N	N	Y	Y
SINA	Y	Y	Y	Y	N	N	N
DSWare	Y	N	Y	N	N	N	N
TikiriDB	Y	Y	Y	Y	N	Y	N
TinyDB	Y	Y	Y	Y	N	Y	N
NoSQL	Y	Y	Y	N	N	Y	N
Modular programming approach							
Impala	Y	N	Y	Y	N	Y	Y
Agilla	Y	Y	Y	Y	N	Y	Y
Application-driven approach							
MiLAN	Y	Y	Y	N	Y	Y	Y
Message-oriented middleware							
Mires	Y	Y	Y	N	N	Y	N

Y: Fully supported, Y-: Partially supported, N: Not supported, N-: Little support

Table 1. Comparison of middleware systems for WSNs (continued).

Middleware system	Security	QoS	Openness	Scalability	Mobility	Heterogeneity	Usability
Virtual machine–based approach							
Mate	Y	N	Y	Y	Y	Y-	Y
MagnetOS	N	N	Y	Y	Y	Y-	Y
SwissQM	Y	N	Y	Y	Y	N-	Y
Database-based approach							
Cougar	N	N	N-	N-	N-	N-	Y
SINA	N	N	N-	N-	N-	N-	Y
DSWare	N	N	Y-	Y-	N-	N-	Y
TikiriDB	N	N	N-	N-	N-	N-	Y
TinyDB	N	N	Y-	Y-	Y-	Y-	Y
NoSQL	N	N	Y-	Y-	N-	N-	Y
Modular programming approach							
Impala	Y	N	Y	Y	Y	N-	Y
Agilla	N	N	Y	Y	Y	Y	Y
Application-driven approach							
MiLAN	N	Y	Y	Y	N-	N-	Y
Message-oriented middleware							
Mires	N	N	Y	Y	Y-	Y-	Y

Y: Fully supported, Y-: Partially supported, N: Not supported, N-: Little support

Table 2. Comparison of the statistics collected from Table 1.

Middleware system	Fully supported (Y)	Partially supported (Y-)	Not supported (N)	Little support (N-)
Virtual machine–based approach				
Mate	9	1	4	0
MagnetOS	10	1	3	0
SwissQM	10	0	3	1
Database-based approach				
Cougar	6	0	4	4
SINA	5	0	5	4
DSWare	3	2	7	2
TikiriDB	6	0	4	4
TinyDB	6	4	4	0
NoSQL	5	2	5	2
Modular programming approach				
Impala	10	0	3	1
Agilla	11	0	3	0
Application-driven approach				
MiLAN	10	0	2	2
Message-oriented middleware				
Mires	7	2	5	0

Acknowledgements. The authors wish to thank the editor and anonymous referees for their helpful comments for improving this paper. This research was supported in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2013R1A1A2011744). Correspondence should be addressed to Dr. Sangman Moh.

References

[1] M. M. Molla and S. I. Ahamed, A survey of middleware for sensor network and challenges, *Proc. of ICPP Workshops*, (2006), pp. 223-228.

[2] S. Hadim and N. Mohamed, Middleware for wireless sensor networks: A survey, *Proc. of 1st Int. Conf. on Communications System Software and Middleware (Comsware 06)*, New Delhi, India, Jan. 8-12 (2006), pp. 1-7.

[3] S. Hadim and N. Mohamed, Middleware: middleware challenges and approaches for wireless sensor networks, *IEEE Distributed Systems* (2006), Vol. 7, No. 3, pp. 1-23.

[4] M. M. Wang, J. N. Cao, J. Li, and S. K. Dasi, Middleware for wireless sensor networks: A survey, *Journal of Computer Science and Technology* (2008), Vol. 23, No. 3, pp. 305-326.

[5] P. Levis and D. Culler, Mate: A Tiny Virtual Machine for Sensor Networks, *Proc. of 10th Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)* (2002), pp. 85-95.

[6] R. Barr, J.C. Bicket, D.S Dantas, B. Du, T.W.D. Kim, B. Zhou, and E.G. Sirer, On the Need for System Level Support for Ad hoc and Sensor Networks, *Operating Systems Review* (2002),Vol. 36, No. 2, pp. 1-5.

[7] R. Mueller, G. Alonso, and D. Kossmann, SwissQM: Next Generation Data Processing in Sensor Networks, *Proc. of 3rd Biennial Conf. on Innovative Data Systems Research (CIDR)* (2007), pp. 1-9.

[8] P. Bonnet, J. Gehrke, and P. Seshadri, Towards sensor database systems, *Proc. of 2nd Int. Conf. on Mobile Data Management (MDM 01)*, Hong Kong, China (2001), pp. 314-810.

[9] Y. Yao and J. Gehrke, The cougar approach to in-network query processing in sensor networks, *ACM SIGMOD Record* (2002). Vol. 31, No. 3, pp. 9-18.

[10] C. Srisathapornphat, C. Jaikaeo, and C. Shen, Sensor information networking architecture, *Proc. of Int. Workshop on Parallel Processing* (2000), pp. 23-30.

[11] S. Li, S. Son, and J. Stankovic, Event detection services using data service middleware in distributed sensor networks, *Proc. of 2nd Int. Workshop on Information Processing in Sensor Networks (IPSN03)*, Apr. 22-23 (2003), pp. 502-517.

[12] J. R. Boteju and C. I. Keppitiyagama, AskME: A database abstraction for ad-hoc networks, *Proc. of Int. Conf. on Advances in ICT for Emerging Regions* (2011), pp. 121.

[13] M. D. W. S. Mahawaththa and M. D. J. S. Goonetillake, Location aware queries for sensor network, *Proc. of Int. Conf. on Advances in ICT for Emerging Regions* (2011), pp. 2-8.

[14] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and H. Wei, TinyDB: An acquisitioned query processing system for sensor networks, *ACM Trans. on Database Systems* (2005), Vol. 30, No. 1, pp. 122-173.

[15] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, The Design of an Acquisitional Query Processor for Sensor Networks, *ACM SIGMOD Int. Conf. on Management of Data*, New York (2003), pp. 491-502.

[16] T. A. M. C. Thantriwatte and C. I. Keppetiyagama, NoSQL query processing system for wireless ad-hoc and sensor networks, *Proc. of Int. Conf. on Advances in ICT for Emerging Regions* (2011), pp. 78-82.

[17] J. Han, H. E, G. Le, and J. Du, Survey on NoSQL Database, Proc. *of Int. Conf. on Pervasive Computing and Applications* (2011), pp. 363-366.

[18] T. Liu and M. Martonosi, Impala: A middleware system for managing autonomic, parallel sensor systems, *Proc. of PPoPP '03,* San Diego, California (2003), pp. 107-118.

[19] C. L. Fok, G. C. Roman, and C. Y. Lu, Mobile agent middleware for sensor networks: An application case study, *Proc. of 4th Int. Conf. on Information Processing in Sensor Networks (IPSN 05)*, Los Angeles, California (2005), pp. 382-387.

[20] W. B. Heinzelman, A. L. Murphy, H. S. Carvalho, and M. A. Perillo, Middleware to support sensor network applications, *IEEE Network (2004)*, Vol. 18, No. 1, pp. 6-14.

[21] E. Souto, G. Guimaraes, G. Vasconcelos, M. Vieira, N. Rosa, and C. Ferraz, A message-oriented middleware for sensor networks, *Proc. of 2nd Int. Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC 04)*, Toronto, Canada (2004), pp. 127-134.

Received: May 1, 2014