A Survey of System Software for Wireless Sensor Networks

Majid Alkaee Taleghan, Amirhosein Taherkordi, Mohsen Sharifi
Computer Engineering Department
Iran University of Science and Technology
{alkaee, taherkordi}@comp.iust.ac.ir, msharifi@iust.ac.ir

Tai-Hoon Kim
Dept. of Multimedia Engineering
Hannam University, Daejeon, Korea
taihoonn@hannam.ac.kr

Abstract

Wireless Sensor Networks (WSNs) are increasingly used as an enabling technology in a variety of domains. They operate unattended in an autonomous way, are inherently error prone and have limited resources like energy and bandwidth. Due to these special features and constraints, the management of sensor nodes in WSNs has its own unique technical challenges as well. For example, system software for WSNs needs to be specially tailored and additionally support specific Quality of Service (QoS) properties such as context-awareness, application-knowledge, reconfiguration, QoS-awareness, scalability and efficiency. A proper classification and comparison of notable system software for WSNs seems to be in order for the literature on this subject. In doing so, in this paper we try to present a survey on such system software with the objective of classifying them especially with regard to aforementioned QoS considerations in the middleware layer. Some classes are identified: agent-based, service-based, query-based, and event-based ones.

Keywords: Wireless Sensor Networks, Middleware, Operating System, QoS Issues

1. Introduction

Wireless Sensor Networks (WSNs) are increasingly used as an enabling technology in a variety of domains. Like other innovative technologies, WSNs have their own unique constraints, capabilities, complexities, features and specific operational environments such as limited and non-renewable energy and resources, small size, low cost, operation in large numbers in physical environments with dynamic and fault prone conditions [1, 2, 3, 4, 5].

WSNs platform and hardware devices are prepared and built by industry and research communities with specific functionalities and in different forms. An interesting point is that sensor nodes are becoming faster, smaller and more powerful [6, 7];

Since WSNs has its own constraints and specific characteristics, designs of middleware systems and Quality of Service (QoS) support are two main challenges in these networks and are open issues for research to realize sensor networks technology [8]. Due to the unique requirements and constraints of WSNs, system software for WSNs needs to be specially tailored [2, 9]. System software, such as an operating system or a middleware, is a software layer which resides above the raw hardware and realizes WSN goals. It manages and controls resources, and provides requested services to application developers. In fact, system-level software hides the constraints and highlights the capabilities to help developers building WSNs applications easily without being involved in detailed low-level complexities [10].

In the literature, most of the proposed middleware systems have reported the layouts of their middleware systems. Since some of the challenges in WSNs like time synchronization, sensor or event location, task scheduling and task partitioning over sensor nodes, and the support of QoS parameters are under considerations [11], we can suppose some of these as main components of a middleware for WSNs. Efficiency, scalability and dependability can be affect the designs of middleware. Also, the design and implementation of an appropriate middleware layer for fully realizing the capability of sensor technologies and applications is now open [6, 12, 13].

Most of work do not consider the above challenges and only propose their middleware layouts, and have no evaluation and mechanism. Some work mention vague algorithms which entail much overhead. We survey the WSNs' middleware systems and explain some of main characteristics of these middleware systems to match with WSNs properties. The design of a middleware layer for sensor networks that fully meets the challenges is now open to discussion [14].

The remainder of paper is organized as follows. First we review the related work in system software for WSNs, and then present some details about WSNs middleware systems. In Section 3, we classify the

middleware systems and show the comparisons of these middleware systems in Section 4. The paper is concluded with an outline of future work.

2. System Software

WSNs can be programmed in a variety of ways [11]: some of the more popular programming models for WSNs include agent-based, query-based and event-based. However, the special characteristics of WSNs make event-based model a more appropriate paradigm for programming. Asynchronous mode of communication in WSNs, and faulty nature of sensor nodes, directs us to this paradigm.

In the past years, there had been a growing interest in event-driven software architectures. These architectures can nicely address the new challenges in distributed computing environments; challenges such as dynamic reconfiguration, multicasting, and loose decoupling between components.

The use of event-based model and the publish-subscribe style of communication as a core model can be applied in variety of applications. Also, the service-oriented architecture [12, 15] can be integrated with publish-subscribe model to bring new component-based model for middleware of WSNs.

As we mentioned, the more challenging part to realize WSN technology is system software. We overview briefly some of the existing operating systems and middleware systems for WSNs.

2.1 Operating Systems

Operating system acts on a single node and manages the tasks and resources of one node only, but middleware can monitor, manage, and adapt all sensor nodes and can reconfigure nodes to increase system lifetime. At the level of a single sensor which has unreliable and low cost properties, we should consider low cost, simple and energy-aware solutions.

Systems such as TinyOS [16] and MagnetOS [17] are classified as WSNs operating systems. These systems do not consider QoS parameters. TinyOS is a popular component-based operating system and is designed for mote sensor nodes [18]. This operating system considers the constraints in sensor nodes and is written by nesC language. It provides support for communication, multitasking and code modularity [2] and is platform specific. MagnetOS provides the illusion of a single Java virtual machine on the top of a distributed sensor network. It is responsible for code partitioning, placement and automatic migration so that total energy consumption is minimized.

At first sight, it seems that QoS guarantee in operating system of WSNs has its overheads and it is thus not feasible. However, AmbientRT [19] uses real-time scheduling in data delivery applications and shows that real-time scheduling is not very resource hungry and it is feasible in data delivery applications in small devices.

2.2 Middleware Systems

Middleware is a software layer and utilizes network and user applications [20]. The WSNs main goal is to collect and deliver data to applications [12]. Therefore, mechanisms should be to deliver data with required qualities and also, guaranteed the application requirements. The development of middleware for sensor networks, however, places new challenges to middleware developers due to the low availability of resources and processing capacity of sensor nodes.

There are several motivations for designing middleware in WSNs. For example MiLAN states, the gap between the protocol and the application is often too large to allow the protocols to be effectively used by application developers [21]. In WSN technology, it is not on which basis future middleware for WSN can typically be built [6], and the fundamental problem of identifying and developing appropriate middleware for fully realizing the capability of sensor technologies and applications remain to be addressed [9].

In the middleware layer, the management of senor nodes can be applied to the whole network and sensor nodes can be managed according to the interests of user applications (interests like quality parameters). Also, different WSNs' applications can be serviced properly. There are few solutions that both operate in the middleware layer of WSNs and in addition, consider QoS parameters like fault-tolerance and real-time.

Fewer works had been reported to consider QoS parameters in WSNs' middleware. [22] uses services and functions for fault detection without recovery and [7] only uses fault recovery. None of these two works consider energy consumption in their solutions and introduce lots of overhead both in time and resource usage. Other important works are as follows.

MiLAN uses graph theory and presents a mechanism to select the best nodes in a sensor network, but the solution is centralized and no evaluation is presented. MidFusion [23] uses Bayesian theory for selecting the best utilized nodes, but unlike MiLAN it does not need a priori knowledge of the type of sensors. MidFusion and MiLAN both have no standard way to represent application requirements.

F. C. Delicato et al. [12] uses greedy algorithm to choose the best nodes which are relevant to application

requirements, it also uses QoS parameters such as data accuracy and energy-awareness in its evaluation. A cluster-based solution in [9] presents some challenges and mechanisms which are important in middleware design and proposes a virtual machine abstraction. Although it considers QoS requirements in its design, but it lacks any mechanism for satisfying QoS parameters.

Other works at middleware layer, such as [24, 25, 26], do not mention QoS in their design, but WSN applications require minimum QoS [21]. Let us thus conclude that most of the proposed mechanisms have no evaluation and are at the least incomplete; only algorithms for QoS considerations are proposed in their middleware layout.

Today, many large-scale distributed systems (e.g. WSNs) need to connect thousands of systems that are widely distributed and change frequently throughout their lifetime. These challenges motivate the demand for a middleware that can provide loosely coupled and asynchronous communication models for distributed systems (e.g. *publish/subscribe middleware*) [15, 27]. The recent middleware systems such as [12, 15, 27, 28, 29] generally use publish-subscribe paradigm as a mechanism to interact between data providers and data consumers in a loosely coupled way.

We describe the following middleware systems consecutively: MiLAN [21], Agent-Based [24], Cluster-Based [9], Reflective [12], Service-Oriented [15], Flexible [28], Role-Based [30], Mires [25], Cougar [26], Request-Reply [31, 32], Garnet [33], and MidFusion [22]. Afterwards we classify them.

Agent-based had considered component-based layered plug-in architecture that is dynamically reconfigured to suit the requirements of applications and changes in environments. Thus it has replaceable and reusable properties. It uses from TCP/IP for protocol stack and there is no evaluation for this approach. It assumes there is java virtual machine on nodes.

Reflective is a middleware for covering the large spectrum of WSN applications requirements, providing both QoS and context-awareness. It satisfies the needs of handling energy efficiently through web service architecture. The modules like adaptation and communication, is used to allow application to change network.

Role-based had proposed a generic role-based abstraction that models application entities as universal plug-n-play roles and network dynamics as changes in node capabilities. The initial simulation is formalizing the role-mapping framework.

Service-centric [34] is not a middleware, but it is a model for modeling the WSN, middleware and protocols. It had proposed a new service-centric model that focuses on services provided by a WSN and views a WSN as a service provider. It has 5 structural layers and 4 functionality sets, such as application for processing, communication for messaging, management for security and reconfiguration, and generational learning for lifespan of WSN. It consists of data-centric, network-centric and resource centric models.

Mires is a message-oriented middleware in a publish-subscribe communication model. It proposes a message oriented middleware with publish-subscribe communication scheme. It implements its middleware on TinyOS operating system and proposes publish/subscribe service for advertisement and notification. It implements aggregation service and use from GUI for user interaction. It has not evaluated on energy consumption or network performance parameters. It implements aggregation service with related functions, such as min, max, average. The user application subscribes to a subset of the attributes offered by the sensor nodes. A subscribe message is broadcasted through the network until it reaches all the sensor nodes. From this moment, the sensor nodes start to monitor, collect, process and transmit the desired information.

Cougar adopts a database approach where sensor readings are considered as “virtual” relational database tables. TinyDB [35] supports a single “virtual” database table for sensors, where each column corresponds to a specific type of sensor (e.g., temperature, light) or other source of input data (e.g., sensor node identifier, remaining battery power). Reading out the sensors at a node can be regarded as appending a new row to sensors.

Mate [36] as a virtual machine provides a simple programming interface to sensor nodes. It implements its own byte-code interpreter on top of TinyOS.

DSWare [37] is a middleware and provides data services for applications. It supports basic events and compound events for event detection and uses confidence functions to trigger an event occurrence. A complete sensor network application will require a number of additional components besides compound event detection [11]. DSWare does not provide support for this glue code, requiring the user to write low-level code that runs directly on top of the sensor node operating system. DSWare supports only a very basic form of compound events: the logical “and” of event occurrences enhanced by a confidence function.

SensorWare [38] is a scriptable lightweight run-time environment and is optimized for sensor nodes with their limited resources. It has the commands like *replicate*, *send*, *query* and *wait* to use sensor nodes’ readings. SensorWare’s programming paradigm allows the implementation of almost arbitrary distributed

algorithms. Typically, there is no need to change the runtime environment in order to implement particular sensing tasks. However, the low performance of interpreted scripting languages might necessitate the native implementation of complex signal processing functions [11].

In table 1, we classify these middleware systems in service-based, event or continuous based, agent-based, query based and hybrid. Also, it shows the operating systems designed for embedded and sensor networks. This classification is based on the way of satisfying user requirements. In service-based, users use simple or composite services such as sensor types and operations performed on. In event-based or continuous, users program a sensor network to somehow respond when an event occurred or collect data continuously. In query-based, WSN collects data and respond when a query inserted in network. Also, a middleware can respond to users in hybrid manner.

3. Comparison

We compared some of the middleware systems had been reported for WSNs and showed in Table 2. The comparison criterions were context-awareness, application-knowledge, QoS-awareness, scalability, efficiency, and energy-awareness.

In MiLAN [21], network plug-in plays an important role and determines which sets of nodes satisfy all of the application's QoS requirements and handle energy. It continuously adapts the network configuration and has the ability to change network parameters and configuration. It decides how best to configure the network to support applications and QoS to increase network lifetime by using from network specific features.

In Agent-based [24], with component-based architecture and respective interfaces, application can inject filtering components in respect to its current conditions or to obtain interesting data. For increased flexibility and decreased resource-consumption, components for managing sensors that are not operating may be removed from the node. The system can compress serialized agents by using a data compression mechanism before the agents are transmitted over a network. The implementation was not built for performance analysis; it was an experiment in component migration.

In Cluster-based [9], the information about environment and failure, periodically gathered through cluster control protocol and updated in the cluster head. Cluster head is responsible for taking adaptation actions. The goal is to maximize the lifetime of the cluster until the first node fails due to depleted battery, while the real-time constraint of the application is satisfied.

Table 1: Sensor networks system software

Service-Based	Event or Continuous-Based	Agent-Based	Hybrid	Query-Based	Operating System
Reflective [12]	Request-Reply [31-32]	Agent-based [24]	Mate [36]	TinyDB [35]	TinyOS [16]
Service-oriented [15]	DSWare [37]	SensorWare [38]	Role-Based [30]	Cougar [26]	MagnetOS [17]
Service-centric [34]	Mires [25]		MiLAN [21]		AmbientRT [19]
Flexible [28]	Garnet [33]		MidFusion [22]		
			Cluster-based [9]		

Table 2: Comparisons of middleware systems

	Context-awareness	Application-knowledge	Reconfiguration	QoS	Scalability	Efficiency
MiLAN [21]	Yes	-	Yes	Yes	-	Yes
Agent-based [24]	-	Yes	Yes	-	Yes	Yes
Cluster-based [9]	-	Yes	Yes	Yes	Yes	-
Reflective [12]	Yes	-	-	Yes	-	Yes
Role-based [30]	Yes	Yes	Yes	Yes	-	Yes

An intuitive objective function is to minimize the maximal energy dissipation among all sensor nodes during each application period.

In Reflective [12], application parameters are information known by the current application and are defined in terms of the values of sensing data. Whenever a significant change is detected in the context, the middleware searches the corresponding application profile to find out how the system should behave in such configuration. Also, it monitors both network and application execution states and performing a network adaptation whenever it is needed.

In Role-based [30], Applications modeled as consisting of active entities playing various roles and interaction between various roles. Each role requires certain capabilities. Application abstraction specified in terms of service and performance metrics. Role abstractions are for services, rules and relevant protocol stack. Role mapping is done by a way of rules and location information. Network resource management is done by load balancing. Sharing and maintaining cross-layer network information among k-hop neighbors, are for context-awareness. The initial evaluation of this middleware is formalizing the role mapping framework.

4. Conclusion

Wireless sensor networks technology is an exciting new technology which has its own specific characteristics, constraints and operational environments. One of the challenges to realize this technology is system software and specially middleware. The gap between the protocols and the application does not allow the protocols be effectively used by application developers [21] and there is still a long way to go for successful WSN middleware, both in terms of design concepts and of system implementations [25].

Recent middleware systems are service-oriented and often uses publish-subscribe style between service consumers and service providers. This style has specific properties like communication decoupling and anonymity which matched with WSNs. WSNs is a candidate for delivering environmental information to different user applications in a ubiquitous environment.

We classified the WSNs' middleware systems and operating systems, and compared some of these middleware systems based on their essential characteristics like context-awareness and QoS support. There are other properties which should be considered in WSNs middleware systems such as security, fault-tolerancy, real-time, interoperability, flexibility, portability, reusability, extensibility, usability, concurrency, adaptability, scalability and localizability and so on.

Today, research communities are working on middleware systems, but there is no standard to specify user requirements and essential middleware services for a specific domain. Also, the integration of many WSNs can be a challenge too.

References

[1] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: research challenges", Ad Hoc Networks Journal (Elsevier), vol. 2, no. 4, pp. 351–367, 2004.

[2] J. A. Stankovic et al., "Real-time communication and coordination in embedded sensor networks", Proceedings of the IEEE, vol. 91, no. 7, 2003.

[3] C. Chong and S. P. Kumar, "Sensor networks: evolution, opportunities, and challenges", Proceedings of the IEEE, vol. 91, no. 8, 2003.

[4] T. Hori, Y. Nishida, N. Yamasaki and H. Aizawa, "Design and implementation of reconfigurable middleware for sensorized environments", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1845-1850, 2003.

[5] M. Ilyas, I. Mahgoub, "Handbook of sensor networks: compact wireless and wired sensing systems", CRC Press, 2004.

[6] K. Romer, O. Kasten, F. Mattern, "Middleware challenges for wireless sensor networks", ACM SIGMOBILE Mobile Communication Review, vol. 6, no. 2, 2002.

[7] A. Lim, "Support for reliability in self-organizing sensor networks", Proceedings of the 5th International Conference on Information Fusion, IEEE Press, vol. 2, 2002, pp. 973-980.

[8] D. Chen, P. K. Varshney, "QoS support in wireless sensor networks: a survey", Proceedings of the 2004 International Conference on Wireless Networks (ICWN 2004), Las Vegas, Nevada, USA, 2004.

[9] Y. Yu et al., "Issues in designing middleware for wireless sensor networks", IEEE Network Magazine, vol. 18, pp.15-21, 2004.

[10] S. S. Yau, F. Karim, "An adaptive middleware for context-sensitive communications for real-time applications in ubiquitous computing environments", Real-Time Systems, vol. 26, pp. 29-61, 2004.

[11] K. Romer, "Programming paradigms and middleware for sensor networks", GI/ITG Workshop on Sensor Networks, pp. 49-54, Karlsruhe, Germany, 2004.

[12] F. C. Delicato et al., "Reflective middleware for wireless sensor networks", SAC'05, Santa Fe, New Mexico, USA, 2005.

[13] L. B. Ruiz, J. M. S. Nogueira, and A. A. Loureiro. "MANNA: A management architecture for wireless sensor networks", IEEE Communications Magazine, vol. 41, no.2, pp. 116-125, 2003.

[14] S. Hadim, N. Mohamed, "Middleware: Middleware Challenges and Approaches for Wireless Sensor Networks", IEEE Distributed Systems Online, vol. 7, no. 3, 2006.

[15] F. C. Delicato et al., "A service approach for architecting application independent wireless sensor networks", Cluster Computing, Springer Science, pp. 211-221, 2005.

[16] J. Hill et al., "System architecture directions for networked sensors", Proceedings of International Conference on Architectural Support for Programming Languages and Operating systems (ASPLOS), Cambridge, MA, 2000.

[17] A. Boulis, C.C. Han, M. B. Srivastava, "Design and implementation of a framework for programmable and efficient sensor networks", MobiSys, San Francisco, USA, 2003.

[18] J. Hill, D. Culler, "A wireless embedded sensor architecture for system-level optimization", Intel Research IRB-TR-02-00N, 2002.

[19] T. Hofmeijer, et al., "AmbientRT - real time system software support for data centric sensor networks", 2nd International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Melbourne, Australia, 2004.

[20] X. Sun, A. R. Blatecky, "Preface: middleware: the key to next generation computing", Journal of Parallel and Distributed Computing, Vol. 64, no. 6, pp. 689-691, 2004.

[21] W. B. Heinzelman et al., "Middleware to support sensor network applications", IEEE Network Magazine Special Issue, vol. 18, no. 1, 2004.

[22] L. B. Ruiz, I. G. Siqueira, L. B. Oliverira,"Fault management in event-driven wireless sensor networks", 7th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Venezia, Italy, 2004.

[23] H. Alex, M. Kumar, B. Shirazi, "MidFusion: middleware for information fusion in sensor network applications", International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Melbourne, Australia, 2004.

[24] T. Umezawa, I. Satoh, Y. Anzai, "A mobile agent-based framework for configurable sensor networks", Proceedings of the 4th International Workshop on Mobile Agents for Telecommunication Applications, pp. 128-140, 2002.

[25] E. Souto et al., "A message-oriented middleware for sensor networks", Proceedings of the 2nd Workshop on Middleware for Pervasive and Ad-Hoc Computing, Ontario, Canada, 2004.

[26] Y. Yao, J. E. Gehrke, "The cougar approach to in-network query processing in sensor networks", Sigmod Record, vol. 31, no. 3, 2002.

[27] P. Costa, G. P. Picco, S. Rossetto, "Publish-subscribe on sensor networks: a semi-probabilistic approach", Proceedings of the 2nd IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Washington DC (USA), 2005.

[28] F. C. Delicato et al., "A flexible middleware system for wireless sensor networks", Middleware, IFIP, LNCS 2672, pp. 474–492, 2003.

[29] M. Sharifi, M. A. Taleghan, A. Taherkordi, "A middleware layer mechanism for QoS support in wireless sensor networks". To appear in the 5th IEEE International Conference on Networking (ICN), Mauritius, 2006.

[30] M. Kochhal, L. Schwiebert, S. Gupta, "Role-based middleware for sensor networks", Wayne State University, WSU-CSC-NEWS/04-TR01, 2004.

[31] J. Blumenthal et al., "Wireless sensor networks - new challenges in software engineering", Proceedings of the 9th IEEE International Conference on Emerging Technologies and Factory Automation, Portugal, 2003.

[32] F. Golatowski et al., "Service-oriented software architecture for sensor networks", International Workshop on Mobile Computing (IMC), pp. 93-98, 2003

[33] L. Ville and P. Dickman, "Garnet: a middleware architecture for distributing data streams originating in wireless sensor Network", 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03), 2003.

[34] D. Gracanin et al., "A service-centric model for wireless sensor networks", IEEE Journal on Selected Areas in Communication, vol. 23, no. 6, 2005.

[35] S. R. Madden et al., "TAG: a tiny aggregation service for ad-hoc sensor networks", In OSDI 2002, Boston, USA, 2002.

[36] P. Levis, D. Culler, "Mate: a tiny virtual machine for sensor networks", Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), San Jose, CA, 2002

[37] S. Li, S. H. Son, J. A. Stankovic, "Event detection services using data service middleware in distributed sensor networks", In IPSN 2003, Palo Alto, USA, 2003.

[38] R. Barr et al., "On the need for system-level support for ad hoc and sensor networks", Operating System Review, vol. 36, no. 2, pp. 1–5, 2002.