A Survey of Middleware for Sensor Network and Challenges

Mohammad M. Molla and Sheikh Iqbal Ahamed
Marquette University, Milwaukee, Wisconsin
{mmolla, iq}@mscs.mu.edu

Abstract

In recent years, Wireless Sensor Network (WSN) has emerged as a highly important research area. Middleware for WSN facilitates development and deployment of a large number of applications such as smart environments, weather forecasting, bridge monitoring, health applications, etc. for sensor networks. But due to resource constraints, unreliability of wireless networks, and diversity in available sensor hardware, middleware for WSN presents a number of new challenges. In this paper, we try to find out and elaborate various challenges associated with the development of middleware for WSN. We present a comparative study of several existing middleware and how they address those challenges. In doing so, we point out the limitations of present generation middleware for sensor networks. We also illustrate how much more work needs to be done to make middleware for WSN suitable for general purpose usage in real world.

Keywords: Middleware, Sensor Networks, Challenges for Middleware, TinyOS

1. Introduction

The growth in integrated circuits (IC) and wireless technologies has triggered the use of sensor network enormously in recent years. According to recent survey [1], WSN is gaining increasing popularity and moving out of research labs into production and real world deployment.

Wireless Sensor Network (WSN) [2] is a form of ad-hoc network consisting of large number of heterogeneous tiny sensors with communication, processing, and storage capabilities. Applications of WSN [3] include habitant monitoring, environmental monitoring, military surveillance, smart houses, intelligent traffic systems, healthcare, and many more. Sensors for WSN differ so much in terms of hardware platforms that writing an Operating System (OS) that runs on all these platforms is impossible. To hide the underlying platform differences and to decouple the OS from hardware platform, we need middleware. Middleware facilitates scalability, interoperability, deployment, and development of applications. There has been numerous works [4-13] on middleware for handheld devices. Most of those handheld devices use operating systems like Windows CE [14], Palm OS [15], Symbian OS [16], Tiny Linux [17], etc. But here we focus on middleware for sensors, which are much smaller than those handheld devices.

During the past few years, researchers have devoted much effort in designing and developing suitable algorithms and programming paradigms for middleware for sensor networks. As a result of this, we have quite a large number of middleware for sensor networks. Impala [18], Mate [19,20], TinyDB [21,22], Agilla [23], TinyCubus [24], and TinyLime [25] are some of the middleware that we have presented in detail in section 3. Other notable middleware for sensor networks are EnviroTrack [26], Mires [27], Hood [28], Cougar [29], DSWare [30], SINA [31], Smart Messages [32], and MiLAN [33].

The purpose of this paper is to find out the challenges associated with sensor networks. We have also done a comparative study of several state of the art middleware for sensor networks to discover the approaches they take to address various challenges associated with sensor networks and also to know how they have succeeded or failed in doing so.

In section 2, we present the various challenges that should be addressed by middleware for sensor networks to be successfully usable in real world. In section 3, we have presented our comparative study of six present generation middleware with a view to finding their usability and limitations followed by a conclusion section with general findings of our study with future research scope.

2. Middleware Challenges for WSN

WSN along with its benefits brings out many new challenges [34-37]. In this section, we try to

articulate the challenges associated with middleware for WSN.

a) Abstraction Support

WSN consists of large number of heterogeneous sensors. These sensors are developed by various vendors and may have different hardware platforms. Hiding the underlying hardware platforms to offer a homogeneous and holistic view of the network is a major challenge for middleware for WSN.

b) Data Fusion

Sensor nodes are used to collect data from its surroundings. Data collected by various sensor nodes have to be merged or synthesized to form more high-level and easily understandable format or report. Also, communicating this synthesized information to the task issuer (i.e. PDA, Laptop, Cell Phone, etc.) is another major challenge.

c) Resource Constraints

WSN consists of tiny sensors with very small memory, computation power, and battery power. Middleware for sensor networks have to be lightweight to work under limited resource availability.

d) Dynamic Topology

Topology of WSN may vary dynamically due to node mobility, node failure, and communication failure between nodes. Middleware for WSN must be capable of handling this dynamic topology of the network.

e) Application Knowledge

Traditional middleware are designed to support a wide variety of applications across the network. But due to limited resource availability, WSN middleware can not be generalized in this way. Middleware for WSN middleware should integrate application knowledge into the services provided. [34-36] propose some approaches for injecting application knowledge in sensor nodes.

f) Programming Paradigm

Due to resource constraints, dynamic network topology, and difficulties involved in collecting and processing sensor data, programming paradigms for middleware for WSN are quite different from traditional programming styles. [38] addresses a number of programming paradigms for WSN middleware.

g) Adaptability

Middleware for sensor networks must support algorithms that have adaptive performance. Adaptive fidelity algorithms [36] have been developed for this purpose.

h) Scalability

Middleware for sensor networks must be scalable enough in terms of number of nodes, number of users, etc. to operate over long period of time

i) Security

Sensor networks have to handle security issues in data processing and data communication. Since sensor networks are often deployed for sensitive applications like military surveillance, patient monitoring, forecasting systems, etc. data collected and distributed by these sensors will have to be authentic and tamper free. But due to limited resource availability and low computation power, most existing algorithms and security models are not suitable for sensor networks. [39-42] address various security challenges and models for sensor networks.

j) QoS Support

WSN middleware should also address various QoS features – response time, availability, bandwidth allocation, etc. for ensuring reliable service.

3. Related Work

In this section, we present a comparative study of several state of the art middleware for sensor network. In doing so, we have taken substantial help from [37].

Most of the middleware we have studied are built on top of TinyOS [44]. So, before delving into our detailed study about each middleware, we present a short introduction to TinyOS. As mentioned in the official homepage [44] of TinyOS – "TinyOS is an open-source operating system designed for wireless embedded sensor networks." Major features of TinyOS are – Component-based architecture, rapid innovation and implementation while minimizing code size, event driven execution model, and fine-grained power management. The programming language of TinyOS is nesC, which is a modified version of C programming language. TinyOS has been ported to over a dozen platforms and numerous sensor boards. Rest of this section focuses on our survey of various middleware.

a) Impala

Impala [10] is a middleware system developed as part of ZebraNet [11] project. ZebraNet project was undertaken to perform long-term migration study of wildlife.

Impala middleware was designed based on an event-based programming model with code modularity, ease of application adaptability and update, fault-tolerance, energy efficiency, and long deployment time in focus. Application adaptability and application update are two major issues implemented by this middleware. It follows a finite state machine based approach taking into consideration various application parameters to handle the adaptability issue. Application updater of Impala is capable of handling incomplete update, inconsistent update, on-the-fly update while code execution, etc.

Although Impala has data communication support for getting data back to the base station, it does not have any support for data fusion. Its abstraction model does not take heterogeneity of the network into consideration and its application domain is rather simplistic.

b) Mate

Mate [19,20] is a virtual machine for sensor networks which is implemented on top of TinyOS [44]. It hides the asynchrony and race conditions of underlying TinyOS.

Mate has a stack-based architecture [45] with three execution contexts – clock, send, and receive. Mate breaks down the program into small self-replicating capsules consisting of 24 instructions. These capsules are self-forwarding or self-propagating.

Although Mate has a small, concise, resilient, and simple programming model, its energy consumption is high for long running programs. Mate's virtual machine architecture increases security somewhat and it takes care of malicious capsules. But its programming model is not flexible enough to support wide range of applications.

c) TinyDB

TinyDB [21,22] is a query processing middleware system based on TinyOS.

TinyDB provides power-efficient in-network query processing system for collecting data from individual sensor nodes which reduces number of messages that must be sent. This results in reduced energy consumption. It has two different types of messages for query processing – Query Messages and Query Result Messages as described in [46]. It also has Command Messages for sending command to sensor nodes.

While TinyDB provides nice abstraction support and has good aggregation model, it does not provide much functionality as part of middleware service. So most of the services have to be provided in the applications running on top of it.

d) Agilla

Agilla [23] is the "first mobile agent middleware for WSNs that is implemented entirely in TinyOS."

Agilla is a Mobile Agent based middleware with 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. Upto four agents can run on a single sensor node. Since one node can run multiple agents at the same time, multiple applications can be supported on the network simultaneously. To save energy, Agilla can move its agent to bring computation closer to data rather than transmitting data over unreliable wireless network.

Agilla does not have any policy for authenticating or monitoring agent activities. Also, its assembly-like and stack-based programming model makes programs difficult to read and maintain.

e) TinyCubus

TinyCubus [20] is a flexible, adaptive cross-layer framework implemented on top of TinyOS.

Flexibility and adaptation are two major issues behind the design philosophy of TinyCubus. To achieve this, TinyCubus architecture is divided into three parts –

i. Tiny Cross-Layer Framework
ii. Tiny Configuration Engine
iii. Tiny Data Management Framework

Although TinyCubus's flexible architecture allows it to be used in different environments, overheard due to cross-layer design may be prohibitive in some environments. Also, adaptation policies are static and scalability is still not good.

f) TinyLime:

TinyLime [25] is implemented on top of TinyOS exploiting Crossbow's Mote platform. It is an extension of Lime [47].

TinyLime follows an abstraction model based on shared tuple space. This tuple space contains sensed

Table 1. Comparison of middleware

Challenge	Impala [10]	Mate [19, 20]	TinyDB [21, 22]	Agilla [23]	TinyCubus [20]	TinyLime [25]
Abstraction	Y	Y	Y	Y	Y	Y
Data Fusion	N	N	Y	Y	N	Y
Resource Constraints	Y	Y	Y	Y	Y	Y
Dynamic Topology	Y	N	Y	Y	Y	Y
Application Knowledge	N	N	N	N	Y	N
Programming Paradigm	Y	Y	Y	Y	Y	Y
Adaptability	Y	Y	N	Y	Y	N
Scalability	Y	N	N	Y	N	N
Security	N	Y	N	N	N	Y
QoS	N	N	N	N	N	N

data. It supports data aggregation to find more information from collected data. TineLime consists of three main components –

i. The Lime Integration Component
ii. The Mote Interface
iii. The Mote-Level Subsystem

TinyLime however does not have any built in security support. Its programming model is rather one time and does not provide good support for adaptability or scalability.

There are many other middleware for WSN – EnviroTrack [26], Mires [27], Hood [28], Cougar [29], DSWare [30], SINA [31], Smart Messages [32], MiLAN [33]. But none of them has good support of security and QoS. They suffer from similar limitations as the middleware we presented above.

We have presented a comparison of middleware in Table 1, summarizing the challenges addressed by several middleware we studied. As we can see from the comparison table, security and QoS are still two most ignored features in current generation middleware. Considering future deployment of sensor networks for sensitive applications, we should start thinking seriously about supporting security in sensor network middleware. Already, there are many theoretical models and protocols [39-42] developed for this purpose. But in practice, those protocols and models are still ignored by most middleware.

4. Concluding Remarks

In this paper, we have presented several challenges that should be addressed by middleware for sensor networks. We then presented a comparative study of six middleware based on their features and limitations followed by a comparison table. From our survey, we have found that existing middleware take quite ad-hoc approach to address various challenges. Although some challenges are addressed by most middleware, their support is not totally satisfactory. For example, all the middleware address the Resource Constraints challenge, but their approach to solve this challenge is not always scalable and adaptable. This limitation hampers the adaptability and scalability of the entire middleware in the long run. Also, some features like – Application Knowledge, Security, and QoS are ignored by most of the middleware. There is virtually no security and QoS support in any middleware. Programming paradigms followed by different middleware are developed in ad-hoc fashion and in many cases not quite flexible. Much research has to be done to develop some unified approach for abstraction, data fusion, and programming paradigm. Also, since security and QoS play key role in sensitive applications deployed over sensor networks, appropriate protocols and models need to be developed for supporting these features. We are currently working on a middleware µ-MARKS to address the challenges we presented in this paper so that we can overcome the limitations of the middleware we studied in this survey.

5. References

[1] Survey: Wireless Sensor Networking Out of the Lab, Into Production. URL: http://www.millennial.net/newsandevents/pressreleases/050824.asp (accessed in February 2006).

[2] F.L. Lewis. “Wireless Sensor Networks,” URL: http://arri.uta.edu/acs/networks/WirelessSensorNetChap04.pdf (accessed in February 2006).

[3] P.J. Marron. “Middleware Approaches for Sensor Networks,” *University of Stuttgart, Summer School on WSNs*

and Smart Objects Schloss Dagstuhl, Aug. 29th – Sep. 3rd, 2005. URL: http://www.vs.inf.ethz.ch/events/dag2005/program/lectures/marron-2.pdf (accessed in February 2006).

[4] M. Sharmin, S. Ahmed, and S. I. Ahamed. "MARKS (Middleware Adaptability for Resource Discovery, Knowledge Usability and Self-healing) for Mobile Devices of Pervasive Computing Environments, " To appear in the *Third International Conference on Information Technology : New Generations (ITNG 2006)*, April, 2006, Las Vegas, Nevada, USA.

[5] The Oxygen Project. URL: http://oxygen.lcs.mit.edu/overview.html (accessed in February 2006).

[6] Project Aura. URL : http://www-2.cs.cmu.edu/~aura/ (accessed in February 2006).

[7] J. P. Sousa, and D. Garlan. "Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments," *Software Architecture: System Design, Development, and Maintenance (Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture)*, August 25-31, 2002, pp. 29-43.

[8] LIME. URL:http://www.cs.rochester.edu/u/murphy/4.pdf (accessed in May 2005).

[9] C. Mascolo, L. Capra, S. Zachariadis, and W. Emmerich. "XMIDDLE: A Data-Sharing Middleware for Mobile Computing," *J. Wireless Personal Comm.*, vol. 21(1), Apr. 2002, pp. 77-103. URL : http://www.cs.ucl.ac.uk/staff/s.zachariad/ papers/ mw.pdf (accessed in May 2005).

[10] P. Wyckoff, S. W. McLaughry, T. J. Lehman, and D. A. Ford. "T Spaces," *IBM Systems Journal*, 1998, pp. 454–474. URL : http://www.research.ibm.com/ journal/ sj/ 373/wyckoff.html (accessed in May 2005).

[11] TSpace. URL : http://www.cs.berkeley.edu/ ~ravenben/research/tuplespace/tuplespace.PPT (accessed in May 2005).

[12] A. T. Campbell. "Mobiware: QOS-aware middleware for mobile multimedia communications," *Proceedings of the IFIP TC6 seventh international conference on High performance networking VII*, White Plains, New York, United States, 1997, pp. 166 – 183.

[13] M. Eichberg, and M. Mezini. "Alice: Modularization of Middleware using Aspect-Oriented Programming," *Software Engineering and Middleware (SEM) 2004*, Linz, Austria, 20-21 September, 2004. URL: http://www.st.informatik.tu-darmstadt.de/database/publications/data/Alice.pdf?id=103 (accessed in May 2005).

[14] WindowsCE. URL: http://msdn.microsoft.com/embedded/windowsce/default.aspx

[15] PALM. URL: www.palm.com (accessed in February 2006).

[16] Symbian OS. URL: www.symbian.com (accessed in February 2006).

[17] Linux PDA. URL: http://tuxmobil.org/pda_linux.html (accessed in February 2006).

[18] T. Liu, and M. Martonosi. "Impala: A Middleware System for Managing Autonomic, Parallel Sensor Systems," *Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming*, 2003, pp. 107 – 118.

[19] P. Levis, and D. E. Culler. "Maté: a Tiny Virtual Machine For Sensor Networks," *Architectural Support for Programming Languages and Operating Systems*, 2002. URL: http://www.cs.berkeley.edu/~pal/pubs/mate.pdf (accessed in February 2006).

[20] B. Blum. "Mate – VM for Sensor Nets." URL: http://zoo.cs.yale.edu/classes/cs434/readings/papers/Mate.ppt (accessed in February 2006).

[21] TinyDB Project. URL: http://telegraph.cs.berkeley.edu/tinydb (accessed in February 2006).

[22] R. Muller. "A Query Processing Middleware for Sensor Networks," *Master Thesis*, September 30th, 2005. URL: http://www.inf.ethz.ch/personal/muellren/thesis.pdf (accessed in February 2006).

[23] Agilla – A Mobile Agent Middleware for Wireless Sensor Networks. URL: http://www.cs.wustl.edu/mobilab/projects/agilla (accessed in February 2006).

[24] P. J. Marrón, D. Minder, A. Lachenmann, and K. Rothermel. "TinyCubus: An Adaptive Cross-Layer Framework for Sensor Networks," *it - Information Technology*, vol. 47(2), 2005, pp. 87-97.

[25] TinyLime. URL: http://lime.sourceforge.net/info/tinyLime.html (accessed in February 2006).

[26] T. Abdelzaher, B. Blum, D. Evans, et el. "EnviroTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks," *IEEE ICDCS* , March 2004. URL: http://www.cs.virginia.edu/~evans/pubs/icdcs2004.pdf (accessed in February 2006).

[27] E. Souto, G. Guimaraes, G. Vasconcelos, et el. "A message-oriented middleware for sensor networks," *Proceedings of the 2nd Workshop on Middleware for Pervasive and Ad-hoc Computing*, Vol. 77, 2004, pp. 127-134.

[28] K. Whitehouse, C. Sharp, E. Brewer, et el. "Hood: a Neighborhood Abstraction for Sensor Networks," *In Proceedings of ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, Boston, MA, June, 2004. URL: http://www.cs.berkeley.edu/~kamin/pubs/hood04mobisys.pdf (accessed in February 2006).

[29] Cougar Project. URL: http://www.cs.cornell.edu/database/cougar (accessed in February 2006).

[30] DSWare. URL: http://www.cs.virginia.edu/colloquia/event302.html (accessed in February 2006).

[31] Srisathapornphat, C.Jaikaeo, and C.C. Shen. "Sensor Information Networking Architecture," *International Workshops on Parallel Processing*, 2000, pp. 23-30.

[32] P. Kang, C. Borcea, Gang Xu, et el. "Smart Messages: A Distributed Computing Platform for Networks of Embedded Systems," *Special Issues on Mobile and Pervasive Computing, The Computer Journal*, 2004. URL: http://discolab.rutgers.edu/sm/papers/sm03.pdf (accessed in February 2006).

[33] MiLAN Project. URL: http://www.futurehealth.rochester.edu/milan (accessed in February 2006).

[34] K. Römer, O. Kasten, and F. Mattern. "Middleware Challenges for Wireless Sensor Networks," *ACM SIGMOBILE Mobile Computing and Communication Review (MC2R)*, Vol. 6, Issue 4, October 2002, pp. 59 – 61.

[35] M. Wolenetz, R. Kumar, J. Shin, and U. Ramachandran. "Middleware Guidelines for Future Sensor Networks," *First workshop on broadband advanced sensor networks (BaseNets)*, October 2004. URL: http://www-static.cc.gatech.edu/~wolenetz/files/pubs/wolenetz_basenets04.pdf (accessed in February 2006).

[36] Y. Yu, B. Krishnamachari, and V. K. Prasanna. "Issues in Designing Middleware for Wireless Sensor Networks," *IEEE Network Magazine*, January 2004. URL: http://www.im.cju.edu.tw/~cflin/931_R2006/2_07.pdf (accessed in February 2006).

[37] P.J. Marron. "Middleware Approaches for Sensor Networks," *University of Stuttgart, Summer School on WSNs and Smart Objects Schloss Dagstuhl*, Aug. 29th – Sep. 3rd, 2005. URL: http://www.vs.inf.ethz.ch/events/dag2005/program/lectures/marron-2.pdf (accessed in February 2006).

[38] K. Römer. "Programming Paradigms and Middleware for Sensor Networks," *GI/ITG Fachgespraech Sensornetze*, Karlsruhe, 26-27 Feb, 2004. URL: http://www.vs.inf.ethz.ch/publ/papers/middleware-kuvs.pdf (accessed in February 2006).

[39] CS691AA - Security and Privacy. URL: http://www.cs.umass.edu/~dganesan/courses/fall05/slides/CS691AA-SecurityPrivacy.pdf (accessed in February 2006).

[40] S. Ozdemir, and P. Nair. "Wireless Sensor Network Security," *Symposium on Research in Engineering & Applied Sciences (REAS '03,)* Arizona State University, September 4th, 2003. URL: http://www.geocities.com/gremates/REAS_WSNSecurity.pdf (accessed in February 2006).

[41] SPINS: "Security Protocols for Sensor Networks," *Wireless Networks Journal (WINE)*, September, 2002. URL: http://www.ece.cmu.edu/~adrian/projects/mc2001/mc2001.pdf (accessed in February 2006).

[42] D. Wagner. "Security for Sensor Networks: Cryptography and Beyond," *2003 ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2003)*, Invited speaker, October 31, 2003. URL: http://www.cs.berkeley.edu/~daw/talks/SASN03.ppt (accessed in February 2006).

[43] P. Juang, H. Oki, Y. Wang, et al. "Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet," *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*, October 2002. URL: http://cs856.watsmore.net/prez-zebranet.pdf (accessed in February 2006).

[44] TinyOS Project. URL: http://www.tinyos.net (accessed in February 2006).

[45] P. J. Koopman. "Modern Stack Computer Architecture," *System Design and Network Architecture Conference*, 1990. URL: http://www.ece.cmu.edu/~koopman/forth/sdnc90b.pdf (accessed in February 2006).

[46] R. Muller. "A Query Processing Middleware for Sensor Networks," *Master Thesis*, September 30th, 2005. URL: http://www.inf.ethz.ch/personal/muellren/thesis.pdf (accessed in February 2006).

[47] A. L. Murphy, G. P. Picco, and G. C. Roman. "Lime: A Middleware for Physical and Logical Mobility," *21st IEEE International Conference on Distributed Computing Systems (ICDCS '01)*, April, 2001, pp. 524-533.