A Middleware Pattern to Support Complex Sensor Network Applications

Bart Elen, Sam Michiels, Wouter Joosen, Pierre Verbaeten
IBBT-DistriNet, Department of Computer Science, K.U.Leuven
Celestijnenlaan 200A
B-3001 Leuven, Belgium
bart.elen@cs.kuleuven.be

ABSTRACT

Most current sensor middleware solutions focus on delivering a particular subset of typical middleware services. Because of this, they only meet the requirements of a, rather limited, set of applications. It is reasonable to expect that a wide variety of applications will be deployed in an industrial WSN (Wireless Sensor Network) environment (e.g. in a harbor that handles container transport). To support such heterogeneous applications, general purpose middleware solutions are needed. We propose a hierarchical layered middleware pattern to handle this heterogeneity. This middleware pattern combines three recurring themes: application management, data management and network service management. These themes were identified by studying state-of-the-art middleware solutions and generalizing common aspects. Combining these three themes allows constructing a middleware which is more general purpose. This makes it useful for a wide variety of heterogeneous applications that will be used in complex WSN environments.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures; C.2.4 [**Computer-Communication Networks**]: Distributed Systems

Keywords

Wireless sensor networks, middleware, architectures, patterns

1. INTRODUCTION

WSNs are currently used for monitoring active volcano's [11], remote health care [1], structural health monitoring [2], and other applications. Due to the considerable challenges WSNs are facing (highly limited resources, hardware and network heterogeneity, need for self-repairing, etc.) [4] application developers need middleware support to facilitate sensor programming. A wide variety of middleware solutions [8, 3, 5, 6, 10, 9, 7] have been proposed to fulfill this need.

In our opinion, sensor programming will evolve from fixed, single purpose (single application) WSNs, to more complex WSN environments. We are participating in the MultiTr@ns project[1], in which WSNs are used in a multimodal transportation context. Sensor nodes installed inside containers, will here become part of heterogeneous WSNs at different locations (ports, trains, boats, trucks, depots, etc.). A wide variety of applications will run on the WSNs to meet the needs of customers, harbor management, security personnel, customs authorities, etc.

This context does impose new requirements on the used middleware. First of all, it will no longer be feasible to use a specific middleware for each WSN application. A more general purpose middleware solution is needed which offers all needed services to the applications that will be running on these WSNs. This evolution is schematically illustrated in Figure 1. Multiple applications make use of the same service components, making it possible to save resources on the sensor nodes. Secondly, sensor middleware must deal with the evolution and heterogeneity of used technologies on different WSNs. Both the network protocols and the application on the WSNs continuously evolve. In addition, they may differ from one WSN to another because of differences in network properties (size, available infrastructure, node mobility, etc.) as well as local user requirements. A static middleware can not deal with this evolution and heterogeneity. Dynamic middleware is needed, which is able to adapt its services to the network and application environment.

Based on a survey on sensor middleware, we will show that most current WSN middlewares focus on a subset of the following three themes: application management, data management and network service management. By consequence, these middlewares only deliver a limited set of services and are unable to support a wide variety of applications.

We argue that complete middleware solutions are needed which combine the application management, data management and network service management themes. The main contributions of this paper are (1) identification of three recurring themes in the existing middleware solutions, and (2) the proposal of a new middleware pattern that is able to support a wide variety of WSN applications by combining these three themes in one hierarchical middleware architecture.

[1]https://projects.ibbt.be/multitrans

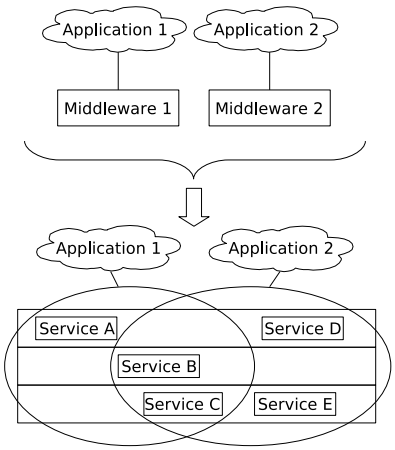


Figure 1: Middleware offering services

The remainder of this paper is structured as follows. In section 2, we give an overview of the services offered by the state-of-the-art sensor middlewares. With the help of a survey we will demonstrate that most middlewares focus on a subset of the three proposed middleware themes, and identify their main shortcomings. In section 3, we propose a new hierarchical middleware pattern which is able to support applications in complex sensor network environments. Section 4 presents conclusions and sketches future work.

2. EXISTING MIDDLEWARES

Software developers for WSNs have to deal with challenges such as managing the limited amount of resources, the need for self-repairing, and environment heterogeneity [4]. Middleware support is needed to help application developers to handle these challenges. A wide variety of middlewares for WSNs have been constructed to meet this need for support.

Based on an extensive study on sensor middleware, we claim that sensor middleware services can be partitioned according to three themes: application management, data management, and network service management. We evaluate existing sensor middlewares on the presence of following main services for each theme:

- Application management theme:
  - **Program global behavior:** It should be possible to program the global behavior of WSNs instead of individual nodes. The middleware has to transform the WSN application into distributed tasks which can be executed on individual sensor nodes.
  - **Application scheduling:** Assign resources to the application. To make this possible the middleware first has to determine the application resource requirements and the availability of resources on the WSN.
  - **Application deployment:** Dynamically deploy the application over the air on the WSN.
- Data management theme:
  - **Local data storage:** Local data storage is needed for information that needs to migrate with the sensor node (e.g. a sensor node in a container traveling from one sensor network to another).
  - **Sensor support:** The middleware should sample the best fitted sensors to deliver the application his needed data. If QoS requirements of the application allow it, the middleware must save energy by lowering the sample rate and by selecting less precise, but also less energy consuming sensors.
  - **Remote data service:** The middleware should offer data on the WSN. This allows remote nodes to request data from the sensor node.
- Network service management theme:
  - **Offer network services:** The middleware should offer the applications an extensible set of network services.
  - **Customize network services:** The middleware should be able to customize the network service implementations on the node to both, service requirements of the applications and the changing local network properties.
  - **Support sensor migration:** The middleware should make it possible for sensor nodes to migrate from one sensor network to another. The needed middleware support includes detecting the presence of reachable WSNs, selecting the best fitted WSN to join, authentication of WSNs, joining WSNs, participating in the local sensor network applications and detecting the sensor network border. Further should the middleware also be able to authenticate sensor nodes that want to join the WSN, accept or reject them and to detect when a node or group of nodes is leaving a sensor network.

Figure 2 summarizes our survey and gives an overview of the main services offered by the current sensor middlewares. In this figure, a '+' identifies a supported service, while a '-' means that the service is partially supported. For each middleware have we identified the theme of focus and marked it gray. We identify three reasons why current sensor middlewares are not suited for complex sensor network environments:

1. None of the current middlewares is offering the complete set of required services. This involves that for each application the appropriate middleware must be selected. When multiple applications are executed on the same sensor node, this may require the usage of multiple middlewares to offer support to all applications. The usage of multiple middlewares implies a considerable increase of used resources.
2. Only TinyCubus [9] and Impala [7] are able to adapt the used network service implementations towards the current network and application environment of the sensor node.

Used symbols '+': supported '-' : partially supported ' ' : not supported	**Program global behavior**	**Application scheduling**	**Application deployment**	**Local data storage**	**Sensor support**	**Remote data service**	**Offer network services**	**Customize network services**	**Support sensor migration**
Abstract regions							-		
Adaptive middl.					+	+			
Agilla	-		+						
AutoSec				-	+		-		
Cougar				-	-	+	-		
DSware				+	-	+	-		
EmStar							-		
Envirotrack	-	+	+		+		-		
Garnet					+	+			
Hood							-		
Impala			+	+				+	
Kairos	+			+			-		
MagnetOS	+	+	+				-		
Maté			+		-		-		
MidFusion					+				
MILAN		+			+	+	+		
Mires					-	+	+		
Regiment	+						-		
Semantic streams					+	+			
SensorWare	-		+						
SINA				+	-	+	-		
Smart messages	-		+						
TinyCubus		+	+	-	-		+	+	
TinyDB				+	-	+	-		
TinyLime				+	-	+	-		

Figure 2: Overview of sensor middlewares and offered services

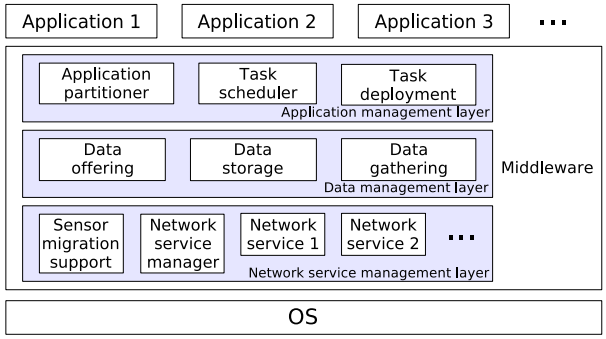


Figure 3: Hierarchical middleware pattern

3. There are still missing parts in the sensor middleware research. An example is the lack of middleware support for the migration of sensor nodes to other sensor networks.

3. NEW MIDDLEWARE PATTERN

In this section we propose a new middleware pattern which provides application developers the necessary support for complex application contexts. As already mentioned in the previous section, this middleware pattern distinguishes itself from the current middlewares by (1) offering a complete set of services and (2) adapting the network services on the node to its current environment. One reason why complete middleware solutions are missing may be its high complexity. It would require expertise in both application management, data management, network service management, energy usage, security, and other domains. Not many are specialized in all these domains. A more easy to realize approach is composing a middleware by using the know-how of multiple groups. It should be possible to combine multiple specialized middleware components to create one, more general purpose, middleware.

To meet this objective, our middleware pattern consists of a hierarchical architecture consisting of three middleware layers, where each layer makes use of the services offered by the layer below. Coherence of components is high within each layer, while coherence between layers should stay limited. Figure 3 shows an overview of this middleware pattern. Each layer contains a number of components which implement a service of the layer. By implementing services in separate components, it will be easier to replace and reuse them. We now zoom in at each layer.

3.1 Applications

We define sensor network applications as the software that is responsible for three concerns: (1) describing when (e.g. each 5 minutes), and where (e.g. in each container) which data (e.g. the temperature) needs to be collected, (2) interpreting the collected data (e.g. a temperature higher then 50 degrees requires special attention), and (3) reacting on it (e.g. send warning to maintenance personnel). WSN applications can be run on a central application server or distributed on the sensor nodes. In the former case the application will not run on the sensor nodes, but makes use of the data management service offered by the WSN.

3.2 Application management layer

The application management layer is responsible for partitioning the sensor network applications into separate tasks, assigning sensor network resources to these tasks and distributing the tasks over the sensor network. Most of its layer implementation will not be on the sensor nodes but on the application deployment server of the sensor network.

3.3 Data management layer

The data management layer is responsible for offering, gathering, and storing data. The local and remote applications indicate their information needs to the data management layer. This layer gathers the needed data by sampling the local sensors and by cooperating with other sensor nodes in the neighborhood. Further the data management layer can save energy by allowing multiple applications to benefit from the same sensor readings.

3.4 Network service management layer

The network service management layer offers a wide variety of network services (routing, data aggregation, distributed position determination, time synchronization, etc.) to the upper layers. This layer is the only network and application dependent middleware part. It deals with both the sensor network and application heterogeneity. The network service management component realizes this by adapting the network service implementations towards the local sensor network properties, the local infrastructure availability, and the service requirements of the applications. Further the network layer gives all support needed for joining and leaving the sensor network (detecting WSNs, authenticating WSNs, detecting WSN borders, etc.). The services in the middleware need to be placed in separate, replaceable components to make it feasible to replace them.

3.5 Interfaces

As already mentioned, most existing middleware solutions focus on a single theme. This makes it possible to combine middleware solutions with different themes to construct middlewares able to meet the requirements of a wide variety of applications. The usage of standardized interfaces is here required. For each layer should be specified which services the layer offers and how their interfaces look like.

4. CONCLUSIONS AND FUTURE WORK

This paper identifies three recurring sensor middleware themes based on a survey of state-of-the-art sensor middleware and it proposes a new layered middleware pattern that combines the three identified themes necessary to deliver the needed application, data and network service management support. This pattern should make it possible to combine middleware solutions to form one, more general purpose, middleware.

The proposed middleware pattern offers four advantages compared to most current middleware solutions: First, resources are saved since only a single middleware solution is needed for all applications. Multiple applications can here benefit from the same service components. Secondly, the middleware is able to adapt the used network service implementations towards the current network and application environment. Thirdly, it allows integrating different expertises by combining middleware layers. Fourthly, code reuse is supported by implementing each middleware service in a separate component.

Concerning future work, we will refine the middleware pattern in an industrial case study on multimodal container transport. We have to complete the services offered by each layer and standardize their interfaces. Further we will build a proof-of-concept prototype of our middleware pattern.

5. REFERENCES

[1] http://www.hoise.com/vmw/05/articles/vmw/lv-vm-04-05-39.html.

[2] K. Chintalapudi, J. Paek, N. Kothari, S. Rangwala, J. Caffrey, R. Govindan, E. Johnson, and S. Masri. Monitoring Civil Structures with a Wireless Sensor Network . IEEE Internet Computing, March/April 2006.

[3] R. Gummadi, O. Gnawali, and R. Govindan. Macro-programming wireless sensor networks using *kairos*. In *DCOSS*, pages 126–140, 2005.

[4] S. Hadim and N. Mohamed. Middleware: Middleware challenges and approaches for wireless sensor networks. *IEEE Distributed Systems Online*, 7(3), 2006.

[5] W. B. Heinzelman, A. L. Murphy, H. S. Carvalho, and M. A. Perillo. Middleware to support sensor network applications. *IEEE Network*, 18(1):6–14, 2004.

[6] P. Levis and D. Culler. Maté: a tiny virtual machine for sensor networks. In *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pages 85–95, New York, NY, USA, 2002. ACM Press.

[7] T. Liu and M. Martonosi. Impala: a middleware system for managing autonomic, parallel sensor systems. In *PPoPP '03: Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 107–118, New York, NY, USA, 2003. ACM Press.

[8] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.

[9] P. J. Marrón, A. Lachenmann, D. Minder, J. Hähner, R. Sauter, and K. Rothermel. TinyCubus: A flexible and adaptive framework for sensor networks. In *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN 2005)*, pages 278–289, January 2005.

[10] M. Welsh and G. Mainland. Programming sensor networks using abstract regions. In *NSDI*, pages 29–42, 2004.

[11] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2):18–25, 2006.