# MidSHM: A Middleware for WSN-based SHM Application using Service-Oriented Architecture

Yuvraj Sahni[*], Jiannong Cao, Xuefeng Liu

*Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong*

**Abstract**

Wireless sensor network (WSN) has been used for structural health monitoring (SHM) application for a long time. However, SHM domain experts lack the knowledge of low-level network issues and cannot develop an application with high efficiency. Middleware hides low-level network and hardware related issues by providing programming abstractions for an application developer which makes it easier to develop an application. In this paper, we survey different middleware approaches and propose MidSHM, a middleware for WSN-based SHM applications using the service-oriented architecture (SOA). SOA uses loosely coupled services that can be reused flexibly to develop different applications. MidSHM is a three-layered middleware with each of the layers containing various services that address issues such as in-network processing, fault tolerance, dynamicity, quality of service etc. We describe operations of various services and use two application examples to illustrate the usability and flexibility of MidSHM. We also compare MidSHM with other SOA-based WSN middleware.

*Keywords:* middleware, structural health monitoring, service-oriented architecture, wireless sensor network

## 1. Introduction

In recent times, civil researchers have begun to shift towards using wireless sensor network (WSN) for structural health monitoring (SHM) application due to its cheaper cost and easier deployment benefits. WSN-based SHM systems are deployed for monitoring the damage in civil infrastructures such as buildings and bridges as shown in Figure 1. Civil researchers have the domain knowledge but they are not expert at handling wireless networking issues which is one of the reasons that there are very few practical deployment of WSN-based SHM system such as [1, 2, 3]. Civil researchers are mainly concerned about getting the raw data from buildings/bridges to a centralized server but this approach is not efficient [4]. We can utilize techniques like distributed computing, in-network processing to make the whole system better but it is difficult to apply these techniques on WSN-based SHM systems. WSN uses resources-constraint devices while SHM application is data-intensive as it requires high sampling rate to acquire sensor data. SHM application also has dynamic network topology and it uses computation-intensive algorithms for damage detection. These challenges make it difficult for civil researchers to develop an efficient system.

Middleware layer can compensate for the huge gap between high-level application requirements and low-level hardware and network issues by providing programming abstractions to hide the complexity of low-lying network. Middleware also provides techniques to optimally manage the network resources and satisfy various application requirements [5]. Middleware layer lies between the application layer at top and WSN system layers at the bottom. By utilizing middleware, application developers do need to learn about hardware and software being used or the operating system. WSN application development becomes a lot easier by using middleware.

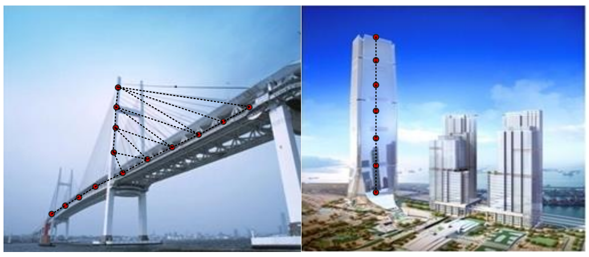

Figure 1: Two SHM systems based on WSN

There are many approaches that have been adopted to develop middleware for WSN but it is difficult to handle all design issues simultaneously. Service-oriented architecture (SOA) approach is becoming popular among researchers for designing middleware due to the nature of services being used for application development. SOA approach can be used to develop middleware that can handle heterogeneity, dynamicity, fault tolerance, quality of service and other issues easily using loosely coupled services. The services used are independent of each other and can be integrated in a customized manner to deal with requirements of different applications. Even new services can

[*]Corresponding author
*Email addresses:* csysahni@comp.polyu.edu.hk (Yuvraj Sahni), csjcao@comp.polyu.edu.hk (Jiannong Cao), csxfliu@comp.polyu.edu.hk (Xuefeng Liu)

be added at run-time to satisfy any new application requirement. SOA can also be used for integration of WSN with the Internet and other networks.

In this paper, we survey various approaches that are used for designing middleware for WSN such as database approach, virtual machine approach, modular approach, message-oriented approach and service-oriented architecture approach. Survey of middleware approaches for WSN has been done previously in [6, 7, 8, 9, 10]. However, in this paper we survey different approaches while focusing mainly on SOA which provides a better understanding of middleware layer for WSN and helps in understanding the benefits of SOA-based middleware over other middleware approaches. Previous studies such as [11] have also done a comparative study of different middleware approaches but they did not consider service-oriented architecture approaches for comparison. This paper also points out some important issues that have not been addressed by existing middleware for WSN-based SHM application.

We also propose MidSHM, an easy-to-use middleware for WSN-based SHM applications that is developed using service-oriented architecture. MidSHM has been designed to deal with issues such as fault tolerance, dynamic application adaptation, resource optimization, quality of service provisioning etc. MidSHM consists of three layers with each layer catering to different issues by using different services. SOA has been used previously for SHM applications but previous designs only deal with some specific issues such as heterogeneity or energy efficiency. There are various issues that have not been explored in much detail such as fault tolerance, quality of service provisioning, support for multiple applications, and integration of different networks. MidSHM overcomes the limitations of previous architectures by using layer level management and integrating the services in those layers to deal with both envisioned and new application requirements [12].

At this stage, we have not fully implemented the MidSHM middleware architecture. We have analyzed services and their operations in different middleware examples and tried to incorporate their functionalities in MidSHM. We have described the operations of various services that are being used in MidSHM. We have adopted many operations from Sensor Web Enablement (SWE) standard [13], and ISHMP services tool suite [14]. We have also described some new services that will address some of the unaddressed issues in existing works. Instead of having a set of services for whole application, we have organized these services in different layers according to their functionality. This makes it easier for an application developer to use them.

The main contributions of this paper are:

1. This paper gives an overview of various middleware approaches being used for WSN. This paper discusses in detail about SOA-based middleware architectures and also classifies them on the basis of applications targeted by these SOA-based middleware architectures.

2. This paper points out application issues that are yet to be addressed by existing middleware architectures for WSN-based SHM application.

3. This paper proposes MidSHM, a middleware for WSN-based SHM application. This paper describes the operations of various services in MidSHM and also uses two application examples to illustrate the flexibility and usability of MidSHM. A comparison study has also been done to compare the functionalities of MidSHM with other SOA-based WSN middleware.

The rest of the papers is as follows. Section 2 discusses different middleware approaches applied for WSN except for SOA-based approach for middleware. Service-oriented architecture approach is described in detail in section 3 using various middleware architecture examples. Section 3 also discusses some unaddressed applications issues by existing middleware architectures for SHM application. Section 4 describes the MidSHM middleware architecture and two application examples enabled by MidSHM. Section 5 is about the implementation of MidSHM architecture and it consists of explanation of various services being used in MidSHM. Section 6 evaluates MIdSHM by comparing its functionalities with other SOA-based middleware for WSN. Section 7 concludes the paper with the conclusion and future work.

## 2. Literature Review

Middleware layer has been used for a long time for decreasing the complexity of WSN application development. There are many approaches for designing middleware for WSN like database approach, virtual machine approach, message-oriented approach, modular approach, application-oriented approach, and service-oriented architecture approach. Table 1 gives a list of different middleware approaches along with their corresponding examples. Description of different middleware approaches along with their limitations has been explained below. This section, however, does not discuss middleware based on SOA approach as it is discussed in detail in section 3.

1. *Database Approach*: This approach considers WSN as a database system where applications can query the data from sensor nodes using structured query language (SQL) like queries. The major drawback with this approach is that it does not provide support for space-time relationship between events. This approach also does not allows rendering of data in real time and provides approximate results. *TinyDB* [15], *SINA* [16], *TinyLIME* [17], *DSWare* [18] are few examples of middleware architectures for WSN system based on database approach. TinyDB is a distributed query processing system that runs on top of TinyOS operating system. It uses acquisitional query processing (ACQP) framework that reduces the power consumption by the inclusion of new query processing techniques and taking benefit of locations and cost of acquiring data [15]. Sensor Information Networking Architecture (SINA) kernel uses a spreadsheet paradigm to query and monitor sensor networks. SINA helps in dealing with data access and aggregation among sensor nodes [16].

Table 1: Various middleware approaches with examples

| Middleware Approaches | Examples |
|---|---|
| Database approach | TinyDB [15], SINA [16], TinyLIME [17], DSWare [18] |
| Message-oriented approach | TinyDDS [19], PS-QUASAR [20], Mires [21] |
| Modular approach | Impala [22], Agilla [23] |
| Application-oriented approach | MiLAN [24], MidFusion [25] |
| Virtual Machine approach | Mate [26], MagnetOS [27] |
| Service-oriented architecture approach | ISHMP Services Toolsuite [14], Servilla [28], OASiS [29], SensorsMW [30], WSN-SOA [31], MidCASE [32] |

2. *Message-oriented approach*: Message-oriented approach uses publish/subscribe communication model to provide asynchronous communication between sender and receiver that are loosely coupled. This approach is very efficient for wireless sensor network systems which require event-based monitoring. Some middleware architectures based on message-oriented approach are *TinyDDS* [19], *PS-QUASAR* [20], *Mires* [21]. TinyDDS is based on Data Distribution service (DDS) specification standardized by Object Management group (OMG) in 2003. TinyDDS is not very lightweight and neither does it provide any provision for reducing energy consumption [19]. PS-QUASAR provides high-level programming model, QoS support (reliability, deadline, priority), and distributed many-to-many exchange of message using multicasting techniques. The drawback with PS-QUASAR is that the routing protocol used is in this middleware architecture requires a lot of memory and also this architecture takes only three QoS factors into account which can be improved [20].

3. *Modular Approach*: This approach is based on the fact that WSNs have dynamic network topology, mobility and require application adaptation to deal with application failure. It is impossible for a single protocol to satisfy the demand of even single application. A monolithic application software that adapts to various envisioned scenarios cannot be used for application adaptation as it is difficult to manage and more prone to errors due to its large size, and it is not suitable for unpredicted future scenarios. A modular approach is required where application is non-monolithic and updates are done on the fly. The drawback with this approach is that it does not allow network heterogeneity because of the nature of its code instruction. *Impala* [22] and *Agilla* [23] are two common examples of middleware architectures based on modular approach. Impala has been built as part of ZebraNet effort [22]. Agilla uses mobile agents for code distribution [23].

4. *Application-oriented approach*: Application-oriented approach provides tight coupling between applications and sensor network. This approach allows applications to specify their requirement in the form of QoS and adjust the network according to application needs. The drawback with this approach is that the middleware designed is not generic. *MiLAN* [24] and *MidFusion* [25] are examples of middleware based on this approach. MiLAN allows applications to specify their QoS requirements in the form of specialized graph and includes state-based changes in application needs. MiLAN configures wireless sensor network according to QoS requirements and maximizes the network lifetime of network [24].

5. *Virtual machine approach*: Virtual machine approach allows applications to be composed of small modules that are distributed through the network using special algorithms. The drawback with this approach is that because of large overhead it is not feasible to execute a large number of executions. *Mate* [26] and *MagnetOS* [27] are two examples based on this approach. Mate is a bytecode interpreter based on virtual machine approach. Mate breaks down the code into capsules of twenty-four instructions that enables installing of wide range of programs using low energy and resource consumption. MagnetOS is an adaptive operating system based on virtual machine approach that is designed for sensor and ad-hoc networks.

## 3. Middleware based on Service-Oriented Architecture

Middleware approaches that have been discussed in section 2 make application development easier by providing programming abstractions and deal with some basic issues such as easier software installation, data aggregation etc. But this is not sufficient because many WSN applications such as SHM require more advanced functions. WSN systems now use heterogeneous hardware and software for application development and deploy multiple applications simultaneously. This makes application development even more difficult and middleware must support flexibility, reusability, heterogeneity, and dynamicity among other features [6].

Service-oriented architecture provides loosely coupled services that are flexible and can be reused to deal with different functionalities of an application. An application is developed by using a group of services that are linked in a particular fashion. The same set of services can be linked in a different manner so as to satisfy the requirements of a different application. There is no need to change the inner details of services in order to deal with new application requirements [14]. However, the major drawback of this approach is that it is not very lightweight. Traditional web services mostly being used for SOA cannot be directly applied for wireless sensor networks because web services have high memory requirement, and high computation and communication cost [33].

### *3.1. Middleware approaches based on SOA*

This subsection discusses some middleware architecture examples based on SOA. There are many other middleware architectures based on SOA like MARINE ([34],[35]), [36], [37] but it is not possible to discuss every middleware example based on SOA. Middleware architectures presented below have been classified in terms of applications targeted. Figure 2 shows three applications based on which different SOA-based middleware are classified.

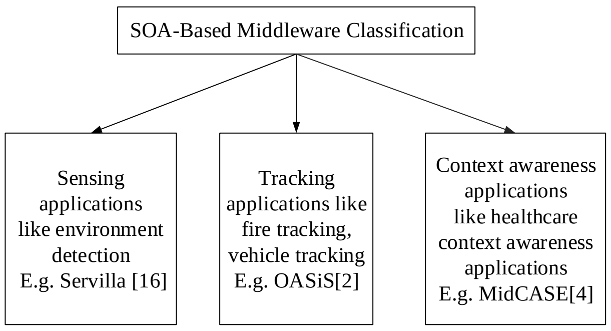
SOA-Based Middleware Classification
Sensing applications like environment detection E.g. Servilla [16]
Tracking applications like fire tracking, vehicle tracking E.g. OASiS[2]
Context awareness applications like healthcare context awareness applications E.g. MidCASE[4]

Figure 2: Application classification for SOA-based middleware

#### Sensing Applications

1. *Illinois Structural Health Monitoring Project (ISHMP) Services Tool suite* [14]: ISHMP services tool suite is an open source toolkit developed at University of Illinois at Urbana-Champaign for SHM application development using smart sensor nodes. This toolkit provides set of services that can be used to implement SHM algorithms for modal analysis and damage detection. Services provided also help in the implementation of middleware SOA-based middleware that facilitates the acquisition and reliable communication of sensor data across the wireless sensor network. The services provided by ISHMP services toolkit can be divided into three categories: *a*) Foundation services, *b*) Numerical services, and *c*) Tools and utilities.

   Foundation services implement functionality needed to support the application. Foundation services also support various services including synchronized sensing of data, precise and accurate time stamping of collected sensor data, reliable communication of data and control commands. Numerical services provide the numerical algorithms necessary to implement the damage detection algorithm such as stochastic damage locating vector (SDLV) on the Imote2s. Application tools are complete applications that are useful in designing, testing, deployment and monitoring of structure for SHM application, while utilities offer a set of basic testing and debugging commands for the existing applications.

2. *Servilla* [28]: Servilla middleware has been designed for heterogeneous WSN by using platform specific services to enable platform-independent applications. This middleware can be utilized to perform sophisticated operations over a large area for a long time by using benefits of both resource-constrained and resource intensive devices to perform various operations. Servilla provides energy efficiency by using novel binding semantics and different service invocation strategies. A new service specification language, ServillaSpec, has been used instead of traditional web services description language (WSDL) to support resource constrained sensor nodes.

   Servilla middleware runs on individual nodes in WSN and it consists of virtual machine (VM) and a service provisioning framework (SPF). Servilla VM is provided by Agilla [23] but it has been modified to meet the needs of middleware architecture. SPF consists of a consumer (SPF-consumer) that discovers and accesses services, and provider (SPF-provider) that advertises and executes services. Servilla has been tested for structural health monitoring application.

3. *Adaptive Servilla* [38]: Adaptive Servilla improves upon Servilla by providing energy-aware provider selection. Adaptive Servilla also increases energy efficiency by exploiting opportunities for sharing service executions and providing dynamic adaptation of provider. Servilla requires developers to manually adjust or specify the type of the bindings but Adaptive Servilla automatically adjusts the bindings based on the expected energy efficiency of the complete system. The process of rebinding to an alternative provider is done dynamically and internally within Adaptive Servilla so the application developer is unaware of this process.

   Adaptive Servilla and Servilla do not support multi-hop network topology and QoS provisioning. While selecting a provider, energy efficiency is considered but there are many other parameters that can also be considered. Adaptive Servilla only provides passive adaption to network topology which leads to high latency so other techniques can be considered to overcome latency issue. Servilla and Adaptive Servilla are enhanced in [39] by including multi-dimensional QoS specifications from each stakeholder into a single value. The value obtained in [39] is then used to determine the best configuration of interactions. Efficacy of Adaptive Servilla has been demonstrated for medical patient monitoring and structural monitoring.

4. *SensorsMW* [30]: SensorsMW is a service-oriented middleware that allows applications to adapt and configure the low-level hardware according to application requirements. It allows applications to specify their QoS requirements using WS-Agreement to monitor and manage Service Level

Agreements (SLA) and dynamically configure the network at run-time. It provides independence from underlying WSN technology. It allows flexibility in delivering data and guaranteeing integration and inter-operability by acquiring data using Web Services. SensorsMW is comprised of four major components: Contracts Creator Component, WSN Gateway, Service Provider, and Data Registry. SensorsMW has been developed for network enterprises and this approach can be used either for periodic measurements or event monitoring. The middleware architecture has been tested for temperature measurement application.

**Tracking Applications**

1. *OASiS* [29]: OASiS is an Object-centric, Ambient-aware, Service-oriented programming framework for WSN application development. OASiS uses programming paradigm that provides Separation of Concerns (SoC). OASiS deals with heterogeneity using service-oriented approach while the dynamic network configuration is provided by ambient-aware middleware. OASiS also supports real-world integration and specification of application specific and network QoS requirements. OASiS framework is very useful to develop data flow applications such as vehicle tracking, fire detection, distributed gesture recognition etc. OASiS uses service graph to describe application data flow. OASiS middleware consists of following services: Node Manager, Object Manager, Service Discovery Protocol, and Composer. This approach has been tested for heat source tracking application.

2. *WSN-SOA* [31]: This middleware architecture provides a multi-level approach that allows SOA to be used at low-capacity nodes without the overhead of XML-based technologies. This approach aims to enable auto-configuration feature at both network and service levels. This architecture classifies nodes in three classes: Full capacity nodes (they use Web services stack and Enterprise Service Buses), Limited capacity nodes (they use device profile for Web services(DPWS)), and Low capacity nodes (they use WSN-SOA stack). WSN-SOA has been implemented using TinyOS. WSN-SOA architecture defines two kinds of services: Management services, sensor/actuator services. The bridging between full capacity nodes and low capacity nodes is done using gateway or bridge. A publish/subscribe data dissemination mechanism using topic-based message filtering system has been provided for more efficiency instead of just using one-one service translation between DPWS and WSN-SOA. WSN-SOA has been tested for surveillance application which involved detecting the intrusions via seismic vibrations and tracking the intruder.

**Context Awareness Applications**

1. *MidCASE* [32] : Middleware Enabling Context-awareness for Smart Environment (MidCASE) is a distributed service-oriented middleware. It has been designed to support programmable application layer consisting of various scenarios with underlying heterogeneous hardware. The middleware provides service in each awareness service domain and the aware process is done by applying rule-based reasoning. SOA allows communication between different factors involved in context-awareness process which was not possible in traditional context awareness systems. MidCASE context awareness system for WSNs consists of three layers: sensor device environment, middleware and application scenario. Middleware component further contains five layers: Hardware abstract, service registry, context-model, awareness and reason, and application presentation layer. MidCASE middleware has been designed for context awareness applications and has been tested for context awareness scenario in healthcare.

### *3.2. Design issues for SOA-based Middleware for SHM application*

Structural health monitoring application involves monitoring the damage in civil infrastructures using hundreds of sensors nodes installed for a long period of time. The data obtained from sensors nodes consists mostly of vibration data from structures and this data is then passed through damage detection algorithms. This whole process must be reliable and robust to withstand various failures in sensor nodes and provide dynamic application adaptation. Middleware based on SOA has been utilized previously for SHM application development [14, 28] but there are a lot of issues which still need to be addressed. Previous approaches generally focus on one of the issues among heterogeneity, dynamicity, in-network processing etc. and a middleware that can handle all the issues still needs to be developed. We have outlined some issues which could be addressed by middleware for SHM application. Addressing such issues would result in development of efficient systems and better utilization of resources.

1. *Quality of Service provisioning in middleware architecture*: QoS is a very important feature to be considered while developing SHM application. There are very few middleware architectures in general which consider QoS issue and there is still a lot of scope for further development in this area especially for SHM application. QoS factors like energy consumption, low latency, reliability, priority, and deadline can be included in middleware architecture to make the system more efficient.

2. *Fault tolerance*: Sensor nodes sometimes get damaged or fail and in some cases, sensors attached to sensor nodes starts giving faulty sensor readings [40]. This issue is very important for SHM application as the device nodes are generally deployed in harsh environments where the possibility of such failure increases. Previous middleware architectures only consider damage node scenario but there is more to fault tolerance than that.

3. *Dynamic service binding*: Dynamic service binding has been used previously to adapt the WSN to network heterogeneity, dynamicity, and other requirements but there is still

a lot of scope for improvement [38]. For e.g. only passive binding has been used in [38] which leads to high latency and new binding techniques can be tried to improve the overall system.

4. *Provision to address the emergency situation*: Most of the times, a node in WSN is either asleep or transmitting ambient data [41]. So, a provision must be there to wake up the nodes in an emergency situation so that emergency event data is recorded and damage can be assessed accordingly.

5. *Damage detection algorithm selection*: There are many, distributed or centralized, damage detection algorithms that have been proposed for SHM application. Such large variety of algorithms can be used for our advantage. Different algorithms can be used at different times according to the data collected or current situation to reduce the total energy consumption. System identification method technique has been swapped in [14] and this approach of swapping different techniques can be extended further. Such kind of algorithm selection or swapping techniques can be helpful in minimizing the consumption of energy as well as other resources.

6. *Provision to deploy multiple applications simultaneously over a heterogeneous network*: Heterogeneous devices are being used to satisfy various demands of multiple applications running simultaneously. A network deployed for SHM can be used for other applications too such as home automation, smart lighting, temperature control etc.

7. *Integration of multiple networks*: A wireless sensor network meant for structural health monitoring application can be integrated with other networks such as Radio-frequency Identification (RFID), the Internet etc. to fully exploit the available resources.

## 4. MidSHM - Proposed Middleware Architecture Based On SOA

The survey of different middleware architectures done in section 2 and 3 points out that SOA approach for middleware is suitable for applications like SHM but there is a need for further improvement. MidSHM, a middleware based on SOA approach has been proposed to deal with WSN-based SHM application issues.

The most important feature of any middleware architecture is that it should be easy-to-use. MidSHM provides a wide range of middleware services which are loosely coupled. These independent services can be easily linked together to develop any application as shown later with the help of two application examples. The main challenge behind this approach is that web services standards being used such as SOAP, WSDL, and UDDI use bulky XMl-messaging format which is not suitable for resource-constraint devices used in WSN [29]. Many techniques have been applied previously to overcome this challenge. OASiS [29] used byte sequence messaging format, Servilla [28] developed a simpler service and task description language while WSN-SOA [31] developed a software architecture and protocol to deal with this challenge.

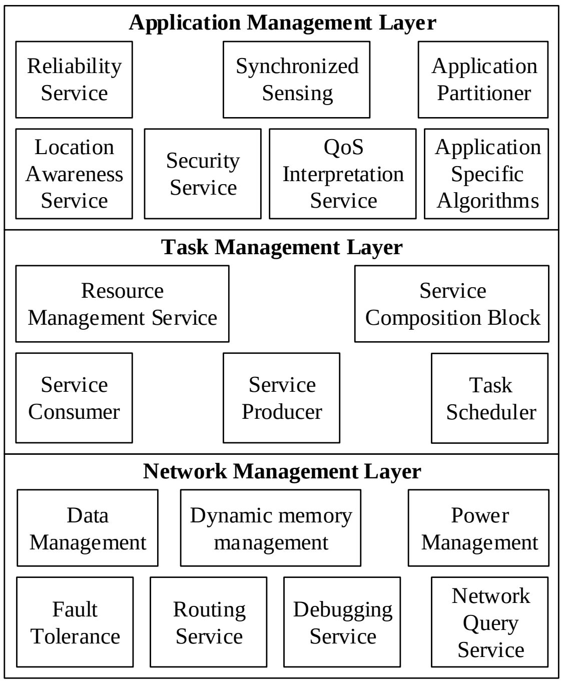

Figure 3: MidSHM middleware architecture

MidSHM can use any of these techniques to deal with the challenge of bulky web services. MidSHM has not been implemented yet and for now only middleware architecture has been described. MidSHM has been designed to provide support for features that are important to SHM application such as resource optimization, dynamic network topology, in-network processing, quality of service, heterogeneity, fault tolerance, real-world awareness, and run-time reconfiguration.

### *4.1. MidSHM description*

MidSHM consists of three layers as shown in Figure 3: Networks management layer, Task management layer, and Application management layer [12]. Three-layer middleware architecture approach has been considered before in [42] [43] but those architectures did not use service-oriented architecture approach for middleware implementation.

Network management layer handles all the network and hardware related issues in wireless sensor network. Task management layer is responsible for managing tasks by using various service modules. Application management layer is the top most layer of middleware and it helps in the management of overall application by dividing each application into multiple tasks. The main function of application management layer is to provide various application-specific services. MidSHM supports the deployment of multiple applications simultaneously. A brief description of various services used in MidSHM middleware architecture is described in Table 2. A detailed description of these services and their operations will be provided later in next section.

### *4.2. Application Examples enabled by MidSHM*

MidSHM middleware shown in Figure 3 has been proposed considering structural health monitoring application but it is flexible and can be used for other applications too. The middleware has not been implemented yet but we use two application examples to illustrate the flexibility and usability of MidSHM. The application examples give an idea of how different services can be utilized to deal with different requirements of an application. Middleware architecture shown in both the examples has been truncated to show only those services that are being used by the application. Although services like resource management service, fault tolerance etc. are very important for the development of WSN system, they have not been used so as to simplify the application. In a practical scenario, these services should not be ignored.

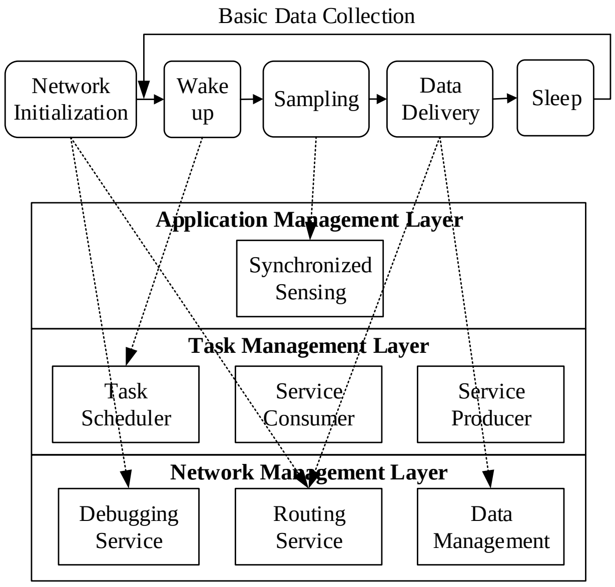

Figure 4: Basic data collection application

Figure 4 shows a basic data collection application. This is a simple application which involves acquiring synchronized data from sensors, performing in-network processing, and finally delivering the data to the destination node. Network Initialization block configures the network, and tests the radio and hardware using debugging service. Wake up block is responsible for setting up the time during which the node is in the active mode for carrying out various tasks. Network Initialization and Wake up block are common for most of the applications but their internal parameter settings may be changed as per the requirement of a particular application. Service Consumer and Service Producer component have been included in middleware architecture because they are main components that carry out calling and execution of services. This application is very basic and it is the building block for most of the WSN-based applications.

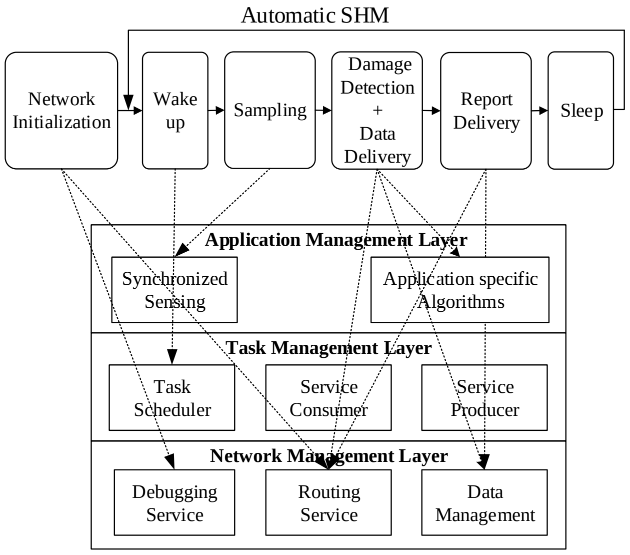

Figure 5: Automatic SHM application

The second application example is Automatic SHM which is shown in Figure 5. Automatic SHM application detects the damage in structure and then automatically delivers the result to base station node for further action. Damage detection and data delivery have been combined into a single block. The damage detection is done within the network using distributed damage detection algorithms for SHM. After final computation, the damage detection result is sent to base station node and this part is represented using Report delivery block in Figure 5. Application specific algorithms block is generic and can be utilized for other applications too. SOA provides the flexibility of adding new services at run-time so new application specific algorithms, as well as new services, can be added to deal with different WSN applications.

Both of the applications presented are very different yet there is not much change required for application development. MidSHM can be utilized for other applications too by using different services depending on the requirement of application. Location awareness service can be used for tracking application that requires knowledge of location of node and its neighbors. In case an application requires certain quality of service factors such as reliability, deadline etc. then QoS interpretation service could be used.

## 5. MidSHM Implementation

MidSHM middleware is based on service-oriented architecture, which comprises of different functionalities that are modeled as loosely coupled services. These loosely coupled services are handled by the service consumer, which is responsible for the discovery and invoking of services, and service producer, which is responsible for providing and executing the services. At this point, we are considering that these services are web enabled, so the MidSHM implementation is depen-

Table 2: Brief description of MidSHM services

| Network Management Layer | |
|---|---|
| Data Management | Reduces the amount of data flowing within the network by using various techniques such as data compression, filtering etc. |
| Power Management | Manages the power consumption of individual node and whole network in order to maximize network lifetime |
| Dynamic Memory Management | Deals with low memory issue in WSN by providing dynamic memory for each task |
| Network Query Service | Queries and sets the parameters related to device node and underlying network including battery power, channel used, transmission power etc. |
| Routing Service | Selects a routing protocol to route the data from one node to another |
| Debugging Service | Used for testing sensor, radio and network configuration before initiating any application task |
| Fault Tolerance | Provides error control and deals with issue of faulty sensor readings |
| **Task Management Layer** | |
| Service Consumer | Responsible for discovering, matching and invoking of services |
| Service Producer | Responsible for publishing and execution of services |
| Task Scheduler | Determines task scheduling according to resource consumption. Three options are available: on-demand, periodic, and event triggered |
| Resource Management Service | Considers parameters from Network query service, QoS interpretation service and other service modules to optimize resource consumption of each task |
| Service Composition Block | In case a single service cannot meet the demands of a task, then this module combines multiple services to deal with the requirement of a particular task |
| **Application Management Layer** | |
| Application Partitioner | Partitions each application into multiple tasks that are then passed to task management layer for further handling |
| QoS Interpretation Service | Interprets the application specific and user specific requirements for any application |
| Location Awareness Service | Determines the absolute and relative location of selected node and its neighbors |
| Reliability Service | Uses different techniques to ensure reliable transmission of data and control commands from source to destination node |
| Application Specific Algorithms | Provides domain specific algorithms required to by WSN application to calculate the final outcome or analyze the data obtained from sensor nodes. For e.g. this can be used to provide algorithms to detect damage in buildings such as ERA, NExT etc. |
| Synchronized Sensing | Responsible for collecting synchronized data from sensors and time stamping. Also synchronizes the global and local clock on each node |
| Security Service | Provides various security features as required by application. For eg: this can be used for setting access control in SHM application |

dent heavily on the choice of implementation of these web services. Web services can be broadly classified into two types: SOAP-based services which are also called big web services, and RESTful web services. A detailed comparison between REST vs SOAP based web services has been done in [44]. Authors in [44] conclude that RESTful web services are easy to build and, provide better flexibility and control for developing web applications but this is only true if enterprise level features such as reliability, security are not required. On the other hand, SOAP based web services have the advantage of better development tools, more alternatives, and support for QoS factors which is useful for developing enterprise level middleware solutions.

Normally it is preferable to use RESTful web services for developing WSN application as RESTful web services are lightweight and thus better suited for resource-constraint devices. But we plan to use SOAP based web services for developing service framework for the MidSHM. The reason behind this is that SHM application is complex and more difficult to develop than usual WSN applications. MidSHM middleware needs to support some high-level features such as fault tolerance, reliability, dynamicity, and many other QoS factors which can be better addressed by using SOAP-based web services. Although SOAP based web services are more suitable for our case, they cannot be used directly as these services use standards such as WSDL, UDDI, WS- Addressing, etc. which are bulky for resource-constraint devices. These services need to be optimized by using techniques presented in [45] [33] such as using optimized data encoding, low-cost binding, implementing improved DPWS architecture etc.

There are two methods for defining the service interface for SOAP based web services: contract first or top-down approach and contract last or bottom-up approach. Using top-down approach, the service interface is defined first using WSDL file and then we use the skeleton java classes to add the required code and create web services. On the other hand, in the bottom-up approach, we start by creating the java method and then automatically generate WSDL file containing the service interface definition. Even though bottom-up approach may be faster and easier for someone new to developing services, the top-down approach is usually recommended for creating web services as it avoids the leakage problem [44]. Figure 6 shows the SOA concept model where service consumer makes a request using XML to service producer and they also communicate using SOAP protocol to access the service description stored in WSDL directory [46].

Currently, we are following top-down approach and therefore we have focused mostly on describing various services and their operations that are used in MidSHM. Service description using WSDL is an important component in implementing web services as shown in Figure 6. We have considered various services and their operations defined in Sensor Web Enablement (SWE) standard [13] for MidSHM. SWE standard framework, developed by Open Geospatial Consortium (OGC) members, consists of the information model and service model that enables implementation of interoperable and scalable service-oriented networks. The information model is made up of different encoding languages that encode observations and metadata received from sensors into a standard format. Encoding of data into a standard format enables faster discovery of information and fosters interoperability between heterogeneous systems. SWE encoding languages that make up the information model are: Observation & Measurement Schema (O&M), Sensor Model Language (SensorML) , and SWE Common Data model [13]. Service model, on the other hand, is made up of four main services that are briefly explained below.

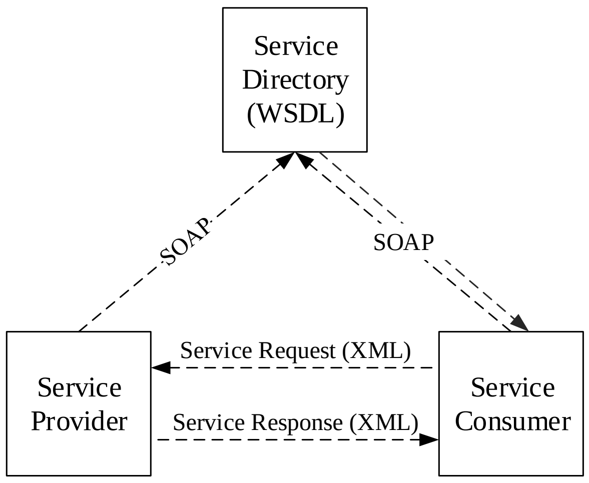

Figure 6: Web service as an implementation of SOA [46]

1. *Sensor Observation Service (SOS)*: This service defines three core operations: GetCapabilities, DescribeSensor, and GetObservations that help in discovering and retrieving the observations and metadata from the sensors. There are some more operations too defined in three other extensions that help in accessing different kinds of data from sensors.

2. *Sensor Planning Service (SPS)*: This service uses various operations that help in maintenance and control of sensor system. This service can be used to obtain all the information related to a requested task including its feasibility status, current status, and also the parameters that are to be set to create a task.

3. *Sensor Alert Service(SAS)*: This service provides publish/subscribe based access to sensor measurements. A user can use this service to subscribe to event notifications. These notifications can be filtered either according to topic or content. Sensor Event Service (SES) [47] which is an enhancement to SAS uses multi-stage content filtering. In the final stage of filtering, the information is filtered using Complex Event Processing (CEP) and Event Stream Processing (ESP).

4. *Web Notification Service (WNS)*: This service provides an interface for asynchronous delivery of messages from other services in SWE standard.

An open source implementation of SWE standard services, done by Sensor Web Community of 52 North, is available at [48]. The implementation of these services can be extended and utilized for implementing the functionality of various operations in MidSHM. SWE standard has been used in developing WSN applications [49] but the services provided by SWE are meant mostly for sensor data and metadata collection. These services cannot be used directly for developing a complex application like SHM. So, we need to extend these services and provide some extra services to deal with SHM application issues. ISHMP services tool suite [14], which has been developed specifically for SHM application, can be used for understanding the required services for SHM application. Although ISHMP has been targeted for SHM application, there are still few issues that have not been considered such as quality of service, fault tolerance for detecting faulty sensor readings etc. We have defined few more services and their operations that were missing both from SWE standard and ISHMP tool suite. Functional description of three layers of MidSHM is shown in Figure 7.

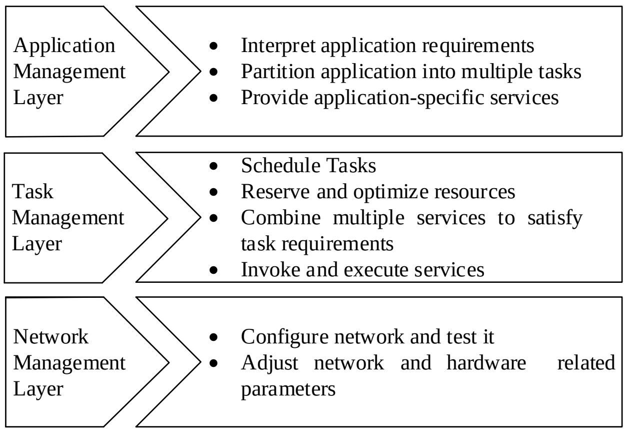
Application Management Layer
Interpret application requirements
Partition application into multiple tasks
Provide application-specific services
Task Management Layer
Schedule Tasks
Reserve and optimize resources
Combine multiple services to satisfy task requirements
Invoke and execute services
Network Management Layer
Configure network and test it
Adjust network and hardware related parameters

Figure 7: Functional description of MidSHM layers

One of the main challenges in implementing such architecture is to incorporate decision making capability at each layer. Application management layer interprets the application requirements and partitions it into multiple tasks. It also determines what services will be used. This decision is made based on the domain knowledge and previous development experiences. Task management layer determines how these services are scheduled and resources are used according to task requirement. Network management layer adjusts network parameters based on the application requirement. Use of domain knowledge and real world knowledge for decision making has been previously considered in iCore project [50]. Another challenge is the high complexity of service interface documents due to large number of operations being used for all the services. Also since these operations are usually imported from one file to another, it sometimes creates problems during processing of web services. Operations of different services being used in MidSHM are explained below in three different subsections.

## *5.1. Application Management Layer*

Application management layer is responsible for managing the overall application. Two important modules required for application management are Application Partitioner and QoS Interpretation Service. These two modules have cognitive functionality that is required for understanding application requirements. Other five service modules present in Application management layer are used for performing application specific functionality.

1. *Application Partitioner*: Application Partitioner partitions the application into multiple tasks and determines what kind of services will be required for application fulfillment. This requires cognitive functionality to understand the application and partition the application on the basis of domain and contextual information available.

2. *QoS Interpretation Service*: This block also has a cognitive functionality that is used for understanding the application requirements from both domain and user perspective. These requirements are useful in partitioning the application into multiple tasks and required service modules.

3. *Security Service*: This service is very important for surveillance applications and many other applications where the data needs to handled carefully. For SHM application, it can be used for setting access control so that data can be accessed by only authorized users. Since this service is not the main priority in SHM applications, we have not discussed it further. Interested readers can refer to [51] to know more about security service in wireless sensor networks.

4. *Location Awareness Service*: This service retrieves the relative and absolute location of a selected node and its neighbors. Two operations can be defined to access the location information: Get Node Location and Get Neighbors. The technique used for obtaining localization information depends on the amount of resources being consumed and quality of service required.

5. *Reliability Service*: uses different techniques to ensure reliable data communication in a WSN by mitigating sensor data loss

6. *Synchronized Sensing*: This service is used for providing time awareness to the device node for synchronizing the sensor data. It performs two different operations:

   a) Synchronize clocks: to synchronize the local and global time of all the clocks

   b) Synchronize sensor data: this operation time stamps the sensor data and synchronizes it. ISHMP services tool suite also has the same service that synchronizes sensor data by resampling it, adjusting sensing start time, and adjusting sampling rate variation with time and in between different sensor nodes.

7. *Application Specific Algorithms*: Application services in ISHMP services tool suite provides many algorithms such

as ERA, NExT, SSI, FDD, SDLV, SDDLV that are useful for detecting and localizing the damage in a structure. Apart from this there are other distributed algorithms such as Energy Efficient clustering [4], and Distributed sensing [52] that can also be used for detection and localization of damage in a structure

### *5.2. Task Management Layer*

Sensor Planning Service (SPS) provides various interfaces that can be used for determining the status of a task and manage it. Operations such as Reserve, Confirm, Update, and Cancel are used for task reservation, confirmation, update, and cancellation respectively. We can obtain the feasibility and parameters required for creating a task using Get Feasibility and Describe Tasking operation. Other operations such as Get Status, Get Task, Get Capabilities, Describe Sensor are useful for obtaining the status and other information related to requested task. The detailed description of the interfaces and their operations can be obtained at [53].

These operations can be used for providing task related information to different services modules present in Task management layer. These service modules are explained below.

1. *Task Scheduler*: Task Scheduler schedules the task and services according to the task information from Application Partitioner and SPS operations. There are usually three types of task scheduling: on-demand, periodic, and event-driven. Event notifications can be adjusted according to content filters defined in SES service [47]. This scheduling is done dynamically and user is unaware of the process. This helps in providing abstractions and results in better resource management.

2. *Resource management service*: Resource management service optimizes the consumption of resources of each task by considering data obtained from network management layer (network and hardware related data), application management layer (application and user specific requirements), and task related data from SPS operations.

3. *Service Composition Block*: This combines various services in order to fulfill the requirements of different tasks. It uses various application specific services present in application management layer and network related services present in network management layer.

Service Consumer and Service provider are the main components in any service-oriented architecture. Service Consumer is responsible for calling and invoking of services while Service Producer provides and executes services. We have only concentrated on service description at this stage and not on these two blocks.

### *5.3. Network Management Layer*

This layer connects the middleware to underlying network and hardware nodes. All the operations related to hardware and network are performed using this layer. These operations are performed by calling various services. This layer is used for performing specific network related tasks such as routing of data to destination nodes, processing of sensor data, managing power consumption in the network by setting low-power mode of devices nodes etc. Network and device-related data obtained using this layer is then utilized by higher layers for resource optimization.

1. *Network Query Service*: The operations in this service, which are shown in Figure 8, are used for providing access to different adjustable parameters of underlying network and device nodes. Some of these operations are common to utility commands provided by ISHMP services tool suite. Sleep operation is used to set the device node in a sleep mode for a specified amount of time. Get List of Nodes retrieves IDs of leaf nodes that are connected to gateway node. The functionality of other operations is clearly understood by their name.

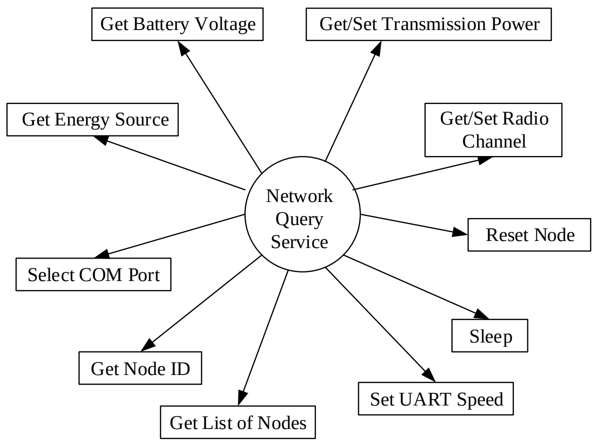

Figure 8: Various operations in Network Query Service

2. *Debugging Service*: Most of the operations in this service are similar to various tools and utilities mentioned in ISHMP services tool suite.

   a) Check Network Topology: used for checking the node connection and wireless topology
   b) Test Radio: used for checking the wireless communication performance by sending the packets between the gateway node and leaf nodes. Input parameters for this are the number of packets to be sent, and IDs of the leaf nodes being tested. On running this command, various performance parameters such as RSSI, LQI, the number of packets sent successfully, and packet error rate on both sides, i.e. on gateway node and leaf node, can be tested.
   c) Test Node: used for checking the data collection and preliminary analysis capability of a node before the start of the actual application. Input parameters for this operation are node ID for testing the node, the number of samples to be collected, and sampling frequency. Once the data is collected it can be viewed and we can perform time series analysis or spectrum analysis on the raw data.

3. *Routing Service*: selects a routing protocol to minimize the energy consumption and provide a fault tolerant network. The routing protocol should be able to handle all three cases: unicasting, multicasting, and broadcasting.

4. *Fault Tolerant Service*: The functionality of this service can be divided into two operations.

   a) Error Control: identify and report errors in transmission and processing of data

   b) Faulty Sensor Reading: use algorithms such as [40] to identify the faulty sensor readings

5. *Data Management Service*: This service manages the data by reducing the amount of data flowing within the network. It uses techniques such as data compression, filtering, or data management using application specific algorithms that convert a large chunk of raw data into small size of useful data which can be easily transmitted further. For e.g. in reference [52], authors have used distributed version of original ERA to reduce the amount of transmission and computation within the network.

6. *Power Management*: This service manages power consumption of whole network by setting the radio transmission power and power mode of nodes. Sometimes the node provides other options for power mode apart from sleep and transmitting mode which can be set accordingly. This decision making is done based on the level of battery power remaining in the node and also its neighbors. The aim of this service is to improve the lifetime of the whole network and not just single node. Some operations which are required to implement this service are:

   a) Get Battery Voltage: to get remaining battery power in a selected node

   b) Get/Set Transmission Power: To get radio transmission power in dBm and set to a required value

   c) Get/Set Power mode: to get the power mode of a selected node and set it accordingly to maximize network lifetime

7. *Dynamic Memory Management*: This service module works along with the service modules from task management layer to determine the amount of memory required for executing a single task. Based on the amount of memory required for a specified task and current remaining memory capacity, this service allocates required memory for a requested task.

## 6. MidSHM Evaluation

The key to solving various issues in MidSHM is separation the issues. Separation of issues is done using partitioning the middleware architecture in three layers [12]. Each layer contains set of services to deal with different requirements. These services are managed by application developer using service consumer and service producer block in task management layer. Services used by the middleware architecture are located within the network which helps in in-network processing of data. Data management module contains different services such as data aggregation, filtering etc. for in-network processing. Issues such as heterogeneity, scalability, run-time reconfiguration are resolved due to independent nature of services used in SOA. Services located on different hardware nodes can be composed together by virtue of being independent. This independent property also allows services to be introduced at run-time allowing dynamic application adaptation. Services can be used on any number of nodes and this allows the network to be scalable.

Routing service can be used to select routing protocols that help in implementing self-healing network. Dynamic service binding can also be used to provide dynamic network topology by allowing nodes to bind to different services in case of device failure. Error control and tolerance against faulty sensor readings are provided by fault tolerance module present in network management layer. QoS interpretation service interprets QoS requirement by different applications while the Network Query Service queries the network status and requirements. These both services pass their reports to task management layer for resource management. Resource optimization is a broad term that includes optimization of energy, memory, computation and communication resources. These resources are centrally controlled by Resource management service which works together with other modules such as data management, power management, and dynamic memory management.

MidSHM provides awareness of time and space associated with data measurement by using location awareness service and synchronized sensing block. Taking real-world awareness in consideration helps in the efficient utilization of service for application development. Services can be deployed on multiple networks and can be used to integrate multiple networks. SOA has been used previously to integrate WSN with the Internet. Application specific issues can be dealt by providing services in application management layer and utilizing application-specific algorithms.

A comparative study of MidSHM with other SOA-based middleware architectures has been done and the results are summarized in Table 3. We have used eight different characteristics to compare different middleware architectures. Characteristics of different middleware architecture have been analyzed and compared with MidSHM. The objective of this comparative study is to find out whether a middleware architecture supports a particular feature or not. *Checkmark* and *Xmark* are used for denoting whether a middleware architecture supports a particular feature or not respectively and in case the feature is supported partially by middleware then *Partial* sign is used. Initial comparison study points out that previous middleware architectures have some drawbacks while MidSHM has the capability to support all design issues relevant to SHM application. The middleware has not been implemented yet so various practical issues that arise while implementing the middleware cannot be addressed at this initial stage.

## 7. Conclusion and Future Work

This paper gives a comprehensive overview on middleware layer for WSN. Various middleware approaches for WSN have

Table 3: A comparative study of various service-oriented architecture based middleware

| | **Resource Optimization** | **Dynamic Network Topology** | **In-Network Processing** | **Quality of Service** | **Heterogeneity** | **Fault Tolerance** | **Real World Awareness** | **Run-time Reconfiguration** |
|---|---|---|---|---|---|---|---|---|
| **ISHMP Services Toolsuite** [14] | ✓ | ✓ | Partial | ✗ | ✗ | ✓ | ✗ | ✗ |
| **Servilla** [28] | Partial | ✗ | Partial | ✗ | ✓ | ✗ | ✗ | ✓ |
| **Adaptive Servilla** [38] | ✓ | ✓ | Partial | ✗ | ✓ | ✗ | ✗ | ✓ |
| **OASiS**[29] | Partial | ✓ | Partial | ✓ | ✓ | Partial | ✓ | ✗ |
| **SensorsMW** [30] | Partial | Partial | Partial | ✓ | ✓ | Partial | ✓ | ✗ |
| **WSN-SOA** [31] | Partial | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | ✓ |
| **MidCASE** [32] | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✓ |
| **MidSHM** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

been surveyed and their drawbacks have been pointed out. This paper points out some of the design issues which are not yet completely addressed by SOA-based middleware for SHM application. Middleware must be flexible and easy-to-use. These two characteristics are achievable using service-oriented architecture approach for middleware. SOA approach uses loosely coupled services for middleware implementation which provide programming abstractions for the application developer and hides low-level issues.

MidSHM, a middleware based on SOA has been proposed to deal with WSN-based SHM application issues. It has been found in initial evaluation study that MidSHM can resolve issues such as Quality of service, fault tolerance, heterogeneity, and dynamicity among other issues discussed in the paper. Two application examples have also been discussed to give an understanding of the usability and flexibility of the proposed middleware architecture.

MidSHM is in its initial development stage and has not yet been implemented. We have only described various services and their operations that are being used in MidSHM. Many of these services have been implemented in SWE and ISHMP services tool suite which makes it easier for us to adapt them. There are still a lot of unknown practical implementation issues that are yet to addressed. The next step will be to complete the implementation of MidSHM and test its performance and usability on real hardware. For future work, a thorough analysis including performance comparison of MidSHM with other middleware approaches will be done after the implementation process is complete.

## Acknowledgment

This work was supported by the the Innovation and Technology Commission of the HKSAR Government [grant number K-BBY1]; the NSFC/RGC Joint Research Scheme [grant number N_PolyU519/12]; the Chinese National 973 project [grant number 2015CB352202]; and NSFC grant of China [grant number 61572218].

## References

[1] S. Jang, B. F. Spencer Jr, Structural health monitoring for bridge structures using smart sensors, Tech. rep., Newmark Structural Engineering Laboratory. University of Illinois at Urbana-Champaign. (2015).

[2] M. Ceriotti, L. Mottola, G. P. Picco, A. L. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta, P. Zanon, Monitoring heritage buildings with wireless sensor networks: The torre aquila deployment, in: Proceedings of the 2009 International Conference on Information Processing in Sensor Networks, IEEE Computer Society, 2009, pp. 277–288.

[3] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, M. Turon, Health monitoring of civil infrastructures using wireless sensor networks, in: Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on, IEEE, 2007, pp. 254–263.

[4] X. Liu, J. Cao, S. Lai, C. Yang, H. Wu, Y. L. Xu, Energy efficient clustering for wsn-based structural health monitoring, in: INFOCOM, 2011 Proceedings IEEE, IEEE, 2011, pp. 2768–2776.

[5] K. Römer, O. Kasten, F. Mattern, Middleware challenges for wireless sensor networks, ACM SIGMOBILE Mobile Computing and Communications Review 6 (4) (2002) 59–61.

[6] N. Mohamed, J. Al-Jaroodi, A survey on service-oriented middleware for wireless sensor networks, Service Oriented Computing and Applications 5 (2) (2011) 71–85.

[7] M.-M. Wang, J.-N. Cao, J. Li, S. K. Dasi, Middleware for wireless sensor networks: A survey, Journal of computer science and technology 23 (3) (2008) 305–326.

[8] L. Mottola, G. P. Picco, Programming wireless sensor networks: Fundamental concepts and state of the art, ACM Computing Surveys (CSUR) 43 (3) (2011) 19.

[9] T. Sheltami, A. Al-Roubaiey, A. Mahmoud, E. Shakshuki, A publish/subscribe middleware cost in wireless sensor networks: A review and case study, in: Electrical and Computer Engineering (CCECE), 2015 IEEE 28th Canadian Conference on, IEEE, 2015, pp. 1356–1363.

[10] S. Hadim, N. Mohamed, Middleware: Middleware challenges and approaches for wireless sensor networks, IEEE distributed systems online (3) (2006) 1.

[11] J. Radhika, S. Malarvizhi, Middleware approaches for wireless sensor networks: an overview, Int J Comput Sci (9) (2012) 6.

[12] Y. Sahni, J. Cao, X. Liu, Midshm: A flexible middleware for shm application based on service-oriented architecture, in: 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE), IEEE, 2016, pp. 126–135.

[13] C. Reed, M. Botts, G. Percivall, J. Davidson, Ogc sensor web enablement: Overview and high level architecture, OpenGIS White Paper OGC 07-165r1, Open Geospatial Consortium Inc. (April 2013).

[14] J. Rice, K. Mechitov, B. Spencer Jr, G. Agha, A service-oriented architecture for structural health monitoring using smart sensors, in: Proceedings of the 14th World Conference on Earthquake Engineering, Beijing, China, 2008.

[15] S. R. Madden, M. J. Franklin, J. M. Hellerstein, W. Hong, Tinydb: an acquisitional query processing system for sensor networks, ACM Transactions on database systems (TODS) 30 (1) (2005) 122–173.

[16] C.-C. Shen, C. Srisathapornphat, C. Jaikaeo, Sensor information networking architecture and applications, Personal communications, IEEE 8 (4) (2001) 52–59.

[17] C. Curino, M. Giani, M. Giorgetta, A. Giusti, A. L. Murphy, G. P. Picco, Tinylime: Bridging mobile and sensor networks through middleware, in: Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on, IEEE, 2005, pp. 61–72.

[18] S. Li, S. H. Son, J. A. Stankovic, Event detection services using data service middleware in distributed sensor networks, in: Information Processing in Sensor Networks, Springer, 2003, pp. 502–517.

[19] P. Boonma, J. Suzuki, Tinydds: An interoperable and configurable, Principles and applications of distributed event-based systems (2010) 206.

[20] J. Chen, M. Díaz, B. Rubio, J. M. Troya, Ps-quasar: A publish/subscribe qos aware middleware for wireless sensor and actor networks, Journal of Systems and Software 86 (6) (2013) 1650–1662.

[21] E. Souto, G. Guimarães, G. Vasconcelos, M. Vieira, N. Rosa, C. Ferraz, J. Kelner, Mires: a publish/subscribe middleware for sensor networks, Personal and Ubiquitous Computing 10 (1) (2006) 37–44.

[22] T. Liu, M. Martonosi, Impala: A middleware system for managing autonomic, parallel sensor systems, in: ACM SIGPLAN Notices, Vol. 38, ACM, 2003, pp. 107–118.

[23] C.-L. Fok, G.-C. Roman, C. Lu, Agilla: A mobile agent middleware for self-adaptive wireless sensor networks, ACM Transactions on Autonomous and Adaptive Systems (TAAS) 4 (3) (2009) 16.

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

[25] H. Alex, M. Kumar, B. Shirazi, Midfusion: An adaptive middleware for information fusion in sensor network applications, Information Fusion 9 (3) (2008) 332–343.

[26] P. Levis, D. Culler, Maté: A tiny virtual machine for sensor networks, in: ACM Sigplan Notices, Vol. 37, ACM, 2002, pp. 85–95.

[27] R. Barr, J. C. Bicket, D. S. Dantas, B. Du, T. Kim, B. Zhou, E. G. Sirer, On the need for system-level support for ad hoc and sensor networks, ACM SIGOPS Operating Systems Review 36 (2) (2002) 1–5.

[28] C.-L. Fok, G.-C. Roman, C. Lu, Servilla: a flexible service provisioning middleware for heterogeneous sensor networks, Science of Computer Programming 77 (6) (2012) 663–684.

[29] I. Amundson, M. Kushwaha, X. Koutsoukos, S. Neema, J. Sztipanovits, Oasis: a service-oriented middleware for pervasive ambient-aware sensor networks, Pervasive and mobile computing journal on middleware for pervasive computing.

[30] G. F. Anastasi, E. Bini, A. Romano, G. Lipari, A service-oriented architecture for qos configuration and management of wireless sensor networks, in: Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on, IEEE, 2010, pp. 1–8.

[31] J. Leguay, M. Lopez-Ramos, K. Jean-Marie, V. Conan, An efficient service oriented architecture for heterogeneous and dynamic wireless sensor networks, in: Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on, IEEE, 2008, pp. 740–747.

[32] Y. Bai, H. Ji, Q. Han, J. Huang, D. Qian, Midcase: a service oriented middleware enabling context awareness for smart environment, in: Multimedia and Ubiquitous Engineering, 2007. MUE'07. International Conference on, IEEE, 2007, pp. 946–951.

[33] N. B. Priyantha, A. Kansal, M. Goraczko, F. Zhao, Tiny web services: design and implementation of interoperable and evolvable sensor networks, in: Proceedings of the 6th ACM conference on Embedded network sensor systems, ACM, 2008, pp. 253–266.

[34] F. C. Delicato, J. M. Portocarrero, J. R. Silva, P. F. Pires, R. P. de Araújo, T. Batista, Marine: Middleware for resource and mission-oriented sensor networks, ACM SIGMOBILE Mobile Computing and Communications Review 17 (1) (2013) 40–54.

[35] W. Li, F. C. Delicato, P. F. Pires, Y. C. Lee, A. Y. Zomaya, C. Miceli, L. Pirmez, Efficient allocation of resources in multiple heterogeneous wireless sensor networks, Journal of Parallel and Distributed Computing 74 (1) (2014) 1775–1788.

[36] F. C. Delicato, P. F. Pires, L. Pinnez, L. Fernando, L. da Costa, A flexible web service based architecture for wireless sensor networks, in: Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on, IEEE, 2003, pp. 730–735.

[37] I. K. Samaras, J. V. Gialelis, G. D. Hassapis, Integrating wireless sensor networks into enterprise information systems by using web services, in: Sensor Technologies and Applications, 2009. SENSORCOMM'09. Third International Conference on, IEEE, 2009, pp. 580–587.

[38] C.-L. Fok, G.-C. Roman, C. Lu, Adaptive service provisioning for enhanced energy efficiency and flexibility in wireless sensor networks, Science of Computer Programming 78 (2) (2013) 195–217.

[39] C.-L. Fok, C. Julien, G.-C. Roman, C. Lu, Challenges of satisfying multiple stakeholders: quality of service in the internet of things, in: Proceedings of the 2nd Workshop on Software Engineering for Sensor Network Applications, ACM, 2011, pp. 55–60.

[40] M. Z. A. Bhuiyan, J. Cao, G. Wang, X. Liu, Energy-efficient and fault-tolerant structural health monitoring in wireless sensor networks, in: Reliable Distributed Systems (SRDS), 2012 IEEE 31st Symposium on, IEEE, 2012, pp. 301–310.

[41] Z. Wang, S. Pakzad, L. Cheng, Sandwich node architecture for agile wireless sensor networks for real-time structural health monitoring applications, in: SPIE Smart Structures and Materials+ Nondestructive Evaluation and Health Monitoring, International Society for Optics and Photonics, 2012, pp. 83450L–83450L.

[42] B. Elen, S. Michiels, W. Joosen, P. Verbaeten, A middleware pattern to support complex sensor network applications, status: published.

[43] Y. Yu, B. Krishnamachari, V. K. Prasanna, Issues in designing middleware for wireless sensor networks, Network, IEEE 18 (1) (2004) 15–21.

[44] C. Pautasso, O. Zimmermann, F. Leymann, Restful web services vs. big'web services: making the right architectural decision, in: Proceedings of the 17th international conference on World Wide Web, ACM, 2008, pp. 805–814.

[45] G. Moritz, F. Golatowski, C. Lerche, D. Timmermann, Beyond 6lowpan: Web services in wireless sensor networks, Industrial Informatics, IEEE Transactions on 9 (4) (2013) 1795–1805.

[46] M. Mokhtary, Sensor observation service for environmental monitoring data.

[47] J. Echterhoff, T. Everding, Opengis sensor event service interface specification, OpenGIS Discussion Paper OGC 08-133, Open Geospatial Consortium Inc. (October 2008).

[48] 52North, Sensor web, `http://52north.org/communities/sensorweb/`, accessed: 2016-06-19.

[49] J. G. Foley, Sensor Networks and Their Applications: Investigating the Role of Sensor Web Enablement, University College London (University of London), 2014.

[50] iCore, Architecture reference model, `http://www.iot-icore.eu/home`, project Number: 287708,Accessed: 2016-06-29.

[51] P. Sakarindr, N. Ansari, Security services in group communications over wireless infrastructure, mobile ad hoc, and wireless sensor networks, IEEE wireless communications 14 (5) (2007) 8–20.

[52] X. Liu, J. Cao, W.-Z. Song, S. Tang, Distributed sensing for high quality structural health monitoring using wireless sensor networks, in: Real-Time Systems Symposium (RTSS), 2012 IEEE 33rd, IEEE, 2012, pp. 75–84.

[53] I. Simonis, J. Echterhof, Ogc sensor planning service implementation standard, OpenGIS Standard OGC 09-000, Open Geospatial Consortium (March 2011).