Open Framework Middleware: An Experimental Middleware Design Concept for Wireless Sensor Networks

Muhammad Sohaib Aslam, Eoin O’Regan, Susan Rea, Dirk Pesch,
Centre for Adaptive Wireless Systems,
Cork Institute of Technology, Cork, Ireland
{muhammad.aslam, eoin.oregan, susan.rea, dirk.pesch} @cit.ie

Abstract

Wireless Sensor Networks (WSN) represent an inherently more complex domain for the design and implementation of middleware when compared to traditional Enterprise systems. Existing examples of middleware within WSN are limited in their scope and are primarily affiliated with influencing the behaviour of nodes only, with some middleware being capable of operating at node and gateway level. We propose an Open Framework Middleware (OFM) that addresses these limitations. It is loosely based on the principles of Model Driven Engineering (MDE) and also incorporates some of the concepts of cloud computing’s ubiquity and pervasiveness. It views the network as a single entity, that is a middleware operating at all levels of the network i.e. node, gateway and control level. It provides distribution, scalability, flexibility and adaptability at deployment, operation and integration levels.

I. INTRODUCTION

Wireless Sensor Networks (WSNs) are wireless networks of low power computing nodes with a range of physical sensing elements but very limited processing capabilities. The nodes’ limited computational and memory resources and limited energy strongly limit the overall performance of the network as a whole. To overcome some of the limitations of such systems, hierarchical architectures are possible where a layer of power limited sensor nodes are supported by a layer of super nodes, which have more processing and energy resources. Such systems use a management layer, which distributes the processing requirements according to computational capabilities of the nodes in the wireless network, where the more powerful nodes compensate for low computing devices. In the classification of middleware paradigms such a management layer can in fact be regarded as a type of middleware. In reality middleware is a complex concept which may be interpreted with reference to the particular system in question. According to Bernstein [1] middleware is defined “as a general purpose service that sits between platforms and applications. By platform, we mean a set of low level services and processing elements defined by the processor architecture and Operating Systems (OSs) API”, however a completely different definition according to [2] defines middleware as follows “middleware layer is equivalent to the presentation layer in the OSI model”. A more appropriate definition within the context of this paper was provided in [2] which defined middleware services as the ability to support the WSN development, maintenance, deployment and execution of WSN applications. In addition [2] also states that “the scope of middleware for WSN is not restricted to the sensor network alone, but also covers devices and networks connected to the WSN”.

In the area of Enterprise networks middleware, a well established concept. One of the main reasons for its success is that it has afforded developers the ability to create systems without having to consider resource limitations to any real degree. This is in contrast to the area of WSNs where resource limitations are the primary exigent consideration.

The existing approaches to middleware design in WSNs are primarily focussed on providing an abstraction of the underlying operating system and to provide an easier mechanism for application development and deployment with some middleware designs being capable of operating at node and gateway level. However, to the best of our knowledge there does not exist a middleware that takes a holistic view of the WSN as a single entity, that is a middleware
operating at all levels of the network i.e. node, gateway and control level, and abstracting the whole network into a single system view.

In this paper we propose a new framework for a flexible, adaptive, integrated and intelligent middleware we refer to as the “Open Framework Middleware (OFM)”. OFM is distributed across all levels of the WSN, collaborating at each level to provide service by incorporating the major characteristics of different middleware classifications such as reflection, adaption and distribution etc. The remainder of this paper is organised as follows: Section 2 briefly discusses related work within the WSN middleware field. Section 3 introduces the OFM and its components. In Section 4 a short diagnostics use case scenario with reference to the framework is provided. Finally Section 5 will expand on the future research objectives for this WSN OFM.

II. RELATED WORK

Middleware technology is not new to WSNs. A number of platforms have been developed for this domain. Existing middleware platforms for WSN can be classified into four categories i.e. Traditional or Classic, Data Centric, Virtual Machines, and Adaptive.

Classic: Classic middleware aims to encapsulate the complexity of the underlying communication and sensing system by providing interfaces. This category of middleware includes TinyLime [3], Impala [4], Mires [5], Hood [6] and TeenyLime [7]. While this middleware category provides interfaces to ease application development for WSN, it does not help with the issue of resource limitations of which application designers still have to be aware. Furthermore, distributed processing across multiple nodes is not an inherent feature of this middleware but needs to be included in applications.

Data Centric: This middleware technology enforces the concept of dealing with sensors as data sources where data can be extracted using SQL like queries. Examples of middleware platforms in this category includes Hourglass [8], IrisNet [9], GSN [10], Cougar [11], SINA [12], DSWare [13], TinyDB [14], and Astrolabe [15]. While it may be useful to view the sensor network as a database, this middleware technology only allows query based applications and therefore limits the scope of the sensor network. Actuation tasks cannot be accommodated by this type of middleware. Furthermore, dynamic retasking of the WSN is not possible with data centric middleware.

Virtual Machines: Virtual Machine (VM) based middleware abstracts the node or network into an execution layer that executes multiple instances of application programs or scripts. Virtual machine based middleware platforms in this area include: Mate [16], VM* [17], DAVIM [18], SwissQM [19], FACTs [20], SmartMessages [21], Agilla [22] and Sensorware [23]. While VM based middleware offers great flexibility in terms of the range of applications it supports, the ability to retask the sensor network in a dynamic fashion, and the level of security it offers by encapsulation application code from OS code, the current VMs have limited instruction sets, memory and code execution performance. Application code is small and therefore limits the range of applications that are possible. As the VM abstracts the underlying OS and hardware, QoS and performance constraints are difficult to meet with VM code.

Adaptive: Adaptive middleware approaches use adaptive fidelity algorithms focusing on the adaptability with examples such as Gridkit [24], AutoSec [25], MILAN [26], Adaptive-Middleware [27], MidFusion [28], TinyCubus [29], TinySOA [30], and RUNES [31]. While this classification of middleware provides a promising approach for the re-configuration of system running on sensor device, it does not comprehend the concept of middleware as a domain comprising of technology, distribution, services and management [32]. Adaptive middleware such as RUNES focus mostly on the aspect of re-configurability and adaption discretely i.e. protocols stack reconfiguration at device level only, which implies that context of network reconfiguration and adaptability is quite restricted. Each of the middleware platforms have specific deficits as identified above, including scalability, QoS limitations, resource constraints and lack of domain knowledge. These constraints limit the overall effectiveness of the WSN and its applications. For further details on the issues outlined above see [33] and [34].

III. OPEN FRAMEWORK MIDDLEWARE

We are currently developing OFM as a theoretical approach to a middleware framework for WSNs. The core concept of this approach revolves around principles of model based
application, interpretation and implementation, knowledge based systems, context-aware/ubiquitous computing systems, intelligent open systems and service based cloud computing. The fundamental approach we propose is to separate and distribute elements of the framework so that it may be capable of altering its overall infrastructure under certain conditions. Furthermore, it provides collections of on-demand and active services distributed among internal domains of the middleware. This provides a structure that can accommodate any middleware infrastructure such as post and subscribe or information advertising mechanism supporting features such as reflection, adaptation, messaging, and event and aspect orientation.

OFM also enforces the concept of division of tasks depending on the processing capabilities of elements such as devices or platforms available at any given level in WSN. This improves the prospects for open scalability, efficiency, adaptability and load distribution. On the contrary, traditional middleware approaches follow rigid and inflexible frameworks, which allow for little change at infrastructure level. In this framework we are focusing on the principle of making the middleware an independent entity with minimum dependency on other components within the WSN by using associations where an association indicates a loose relation with other components through models which will be explained further. As OFM is an intelligent system it can if necessary provide an “ask and do” mechanism for certain relations to be established if required based on QoS requirements. OFM also allows models to describe QoS requirement openly i.e QoS of application or single component or the network as whole can be expressed in models, which become active services to monitor, report and manage QoS in a distributed manner.

In general the purpose of OFM will be as an open system to interpret the specification, instructions and configurations given to it in the form of models and define its actions accordingly to accommodate number of middleware approaches in WSN, thereby removing the constraint of a rigid and immutable implementation of the framework as is the case with traditional middleware approaches. OFM can be generally divided into the following parts based on the purpose and functionality.

• Models.
• Gateway middleware.

Models: OFM models are multi purpose models consisting of data and structure. The data in the models can be code which may influence the behaviour of the middleware or it could be a functional description of a single device in the WSN that may evolve the middleware itself. For example, if we create a sensing model, this will be received by the middleware and will evolve the middleware [gateway, micro] so that it can instigate the functions specified in the sensing model. The structure in the models can be used as a validation and verification mechanism.

Model classification: There are three main model classifications i.e. Infrastructure/Behaviour Selection Inductive Models (ISIM), Operational/Functional Declarative Models (ODM) and Refined Action Models (RAMs). Figure 1 shows the model classification.

A. ISIM: It contains indicator information along with the set of behaviour modifications that may change the infrastructure and behaviour of the gateway middleware.

B. ODM: these are strictly operational models which do not specify any behaviour modification. Currently ODM supports limited elements such as data, events, and actions etc.

C. RAMs: These are created by the gateway middleware services for the micro middleware which runs on super nodes. Super nodes are high end wireless platforms that support low end sensor nodes to create an efficient wireless network. RAMs may contain information for specific super nodes such as identifying the functions of a super node or how super nodes communicate with low end nodes etc.
Gateway middleware: A domain based system comprising of on-demand and active services. Its function is to process models and disseminate the information extracted. It is then deployed to the corresponding domain depending on the model classification i.e. ISIM or ODM. Again taking the example of the sensing model, the gateway middleware receives this model and using services in its domain, the structure and data is extracted, interpreted. It then identifies the internal data [to be used by itself] and the data to be disseminated to super nodes or low computing nodes.

This approach allows to have a number of different specifications for temperature sensing models where each specification has a particular functional requirement. For example, if we have two models of a single function for temperature sensing, each version of the model may have data to be disseminated to a particular domain of middleware [gateway, micro] i.e. version one of the model may only provide a simple interface to the control application and have a single purpose of sensing temperature. The gateway middleware only acts as a communication relay between the control application and the node reporting the information. On the other hand version two of the same model may evolve the system to sense and monitor with reference to a map of specified ranges and provides an interface to the control application with extended functionality.

The following are the key components that comprise the gateway middleware. All components are listed in Figure 2.

**Gateway middleware classification:** OFM gateway middleware is classified into three domains based on its common operations, functions and behaviours.

A. **Operation/Function Domain (OD):** contains components that are vital transactional components. They do not impact on any other domain in the OFM gateway middleware. The components in OD use ODMs to perform requested operations as distributors, communicators, formatters etc. For example **Distributors** are composed of on-demand deployment services. **Verification and Integration Services** execute on-demand validation and integration of data into the OD. **Communication UNITs** are active services which may act as a communication pipeline for applications utilising the WSN.

B. **Infrastructure/Behaviour Domain (ID):** This domain controls the behaviour of the OD. The ISIMs contain indicators which point to the set of behaviour modifiers where each modifier set changes the infrastructure of the gateway middleware provided that the set already exists. Otherwise the ISIMs must have the set description. This change influences the OD by defining how the components in the OD will work e.g. it may define how the communication UNIT in the OD will communicate with the control application and the underlying wireless sensor node network.

C. **Common Operation Domain (COD):** COD is the primary domain which deals with the models. COD processes models into components required by OD and ID. It consists of a **Model Processing UNIT** which processes models and forwards the processed information from the models to the appropriate **Translators**. The Translators interpret the information passed to it and extract meaningful information for the **Micro Knowledgebase**, which is responsible for the storage of meaningful information, or **Service Identifiers** which create services if required and push the information to the corresponding domain.

**Micro middleware:** This element of the OFM inherits its properties and functionalities from the gateway middleware and acts as a computation layer over the super nodes. Effectively the Micro middleware is a scaled down image of the gateway middleware. Unlike the gateway middleware it receives a highly formatted and refined form of the model called RAM.
Figure 2: OFM Layers and Services Classification.
IV. OFM DIAGNOSTICS EXAMPLE

In order to explain the use of the OFM, we outline in the following two example scenarios for the implementation of a Diagnostics model in the OFM. The first Diagnostics model is shown in ‘Scenario A’ of Figure 3 and it aggregates two simple sensing models. The second model in ‘Scenario B’ is a more complex example incorporating a number of other models. For Scenario A, the targeted node(s) simply sense temperature and humidity. The data recorded is relayed through the service running on the Communication UNIT of the Gateway to the level above the WSN where the data is filtered, faults are determined, diagnostic procedures are carried out and fault recovery decisions are made. Depending on the fault recovery decision a Fault Recovery Model may be sent to the middleware and this will then carry out the Fault Recovery functions.

For Scenario B, we have the same two sensing models. This time however, model aggregation forms an overall Diagnostic model in the form of associations where a Fault Detection model associates itself with the Sensing models, Diagnosis model, Fault Recovery and Routing models. The models trigger a functional evolution in the middleware at the OD level. This enables not only sensing of the data but also defines the process for detection, using the fault detection model and a solution in the form of the Routing model, which contains information about alternate routes that can be used as a part of the fault recovery solution. Moreover, multiple solutions can be defined in the form of models. A useful illustration of the concept of aggregation can be seen in Scenario B.

The Diagnosis model is an independent decision making model that implements the core diagnostic functions. It is then associated with the Fault Recovery and Fault Detection models to implement a more comprehensive diagnostics solution.

Figure 3: Diagnostics in OFM

V. FUTURE WORK

As the core concepts and functionality of the OFM have been defined, the next step in this development process is an iterative implementation of the framework. We plan an initial implementation on a server platform and subsequently on the Java SunSPOT [35] platform. Initial experimentation will be based on the scenarios described in Figure 3. One of the major areas in the implementation will be developing the ability to interpret the models. Considering the diagnostic scenario, this will act as a test case for the model development process.

To perform diagnostics we need to identify the stages as fault detection, identification and reconfiguration. Each of these tasks will be defined as a model or series of models with associations defined between these models. To develop the models we need to be aware of the WSN parameters that need to be considered for diagnostics. For example, taxonomy of WSN faults must be specified and their likely causes to facilitate the specification of fault detection and identification models. In addition for reconfiguration we must be aware of the WSN parameters we can modify and also the ranges within which these parameters can function, i.e. the maximum data rate is 250kbps or the device constraints such as temperature sensing capability is from -128°C to +127 for the sensor on e-Demo board available in SunSPOT. For accurate model definition the WSN capabilities and operating conditions must be considered and the OFM is a flexible distributed framework capable of interpreting wide and varying ranges of requirements.

Acknowledgements
The authors acknowledge the support of the Irish Higher Education Authority under the Programme for Research in Third Level Institutions (PRTLI) cycle 4 funded NEMBEES programme and Science Foundation Ireland under grant 07/SRC/11170 in funding part of the work reported in this paper.

Reference:

[22] A. Boulis, C-C Han, R Shea, M. B. Srivastava, SensorWare: Programming sensor networks beyond code update and querying, Pervasive and Mobile Computing, Volume 3 Issue 4, August 2007


[35] SunSPOT, Sun Microsystems, Project SunSPOT , website http://www.sunspotworld.com/docs/