Service-Oriented Middleware Approaches for Wireless Sensor Networks

Nader Mohamed and Jameela Al-Jaroodi

Faculty of Information Technology
UAEU, Al Ain, P.O. Box 17551, UAE
{nader.m, j.aljaroodi}@uaeu.ac.ae

Abstract

While Wireless Sensor Networks (WSN) are starting to become more popular and used for many applications, developing these applications is still not a trivial task due to many technical challenges such as limited hardware capabilities, communication, and energy resources; high heterogeneity; security; and quality of service (QoS) issues. Recently Service-Oriented Middleware (SOM) has become a preferred platform for developing WSN applications over the more traditional development platforms to help address these challenges. SOM logically views WSN as a service provider for user applications. The SOM provides abstractions for the complex underlying WSN infrastructure through a set of services needed by the WSN applications. These services can be data aggregation, adaptation, security, self-organization, reliability, and management services. This paper surveys the current SOM solutions for WSN. The paper discusses the trends and challenges of designing and developing these solutions for WSN.

1. Introduction

WSN are used for many applications such as pollution monitoring, infrastructure protection, disaster prevention, habitat monitoring, healthcare applications, and traffic control. The design and development of such applications must address many challenges dictated by WSN characteristics on one hand and the targeted applications on the other. One of the emerging approaches used for relaxing these challenges is using service-oriented middleware (SOM). Service-oriented computing (SOC), in general, aims to make services available and easily accessible through standardized models and protocols without having to worry about the underlying infrastructures, development models or implementation details.

WSN applications are usually designed to utilize heterogeneous devices such as high-end servers, RFID components, mobile devices, storage devices, sensors, and robots. These devices communicate through a variety of wireless/wired networking technologies. In addition, these devices are controlled by a variety of software components developed by different vendors and implemented using different programming languages and models. Due to the high distribution, dynamic properties, and heterogeneity of the devices, software components, and communication technologies used, implementing such applications is non-trivial. There are integration, scalability, reliability, security, usability, QoS, and operational issues to be considered.

Recently SOC [4] has become a preferred model over the more traditional distributed system development models. It overcomes several problems and challenges these two types have faced. However, SOC is not a perfect solution either. As in any other approach, SOC suffers from serious issues in supporting heterogeneous environments and systems and in providing effective functional and non-functional requirements for the applications. Thus middleware becomes an important approach to use for SOC as well. Thus, SOM has become very important.

Using SOM to support service development and utilization has been advocated by many researchers who call for using SOM over traditional middleware that connects diverse components and systems, and provides multiple-channel access to services, e.g. via mobile and hand held devices and sensor devices over a variety of networks including the Internet, UMTS, XDSL, and Bluetooth [22]. On another aspect, the authors [19] identify the requirements and challenges of SOM for pervasive environments. Among the requirements there are primary requirements such as message exchange, mobility support, security and service description, discovery and coordination. There are also cross-cutting requirements including QoS support, autonomy and context awareness.

In this paper we review and discuss different approaches of SOM for WSN, and present some of the challenges they face. In Section 2 we provide background information and related concepts. Section 3 discusses the general middleware challenges for WSN. The SOM requirements for WSN are discussed in Section 4 and the current SOM approaches for WSN are discussed in Section 5. Section 6 provides a discussion of the SOM challenges for WSN and Section 7 concludes the paper.
2. Background and Concepts

Web services emerged earlier as an easy to use technology to support the concept of software as a service. They provide developers with XML-based methods to access and integrate services available over the Internet and utilize them as part of their own applications. The concept further advanced to result in a larger, more sophisticated framework under SOC. The main advantage of this approach is giving the applications a way to integrate various services available online within the context of the application’s specific domain and using them as needed instead of implementing the whole solution from scratch.

Middleware also offers similar capabilities to integrate and reuse software components on demand. However, it does not support online or on-demand integration easily. It is generally more targeted for in-house solutions rather than service access. Some researchers view SOC as a replacement for middleware since application development will not rely on detailed implementation, but will mainly integrate with available components based on SOC. However, even the SOC architecture itself has grown drastically that it resembles very large scale applications and require careful and well planned design and development. As a result, middleware approaches are used to facilitate the design and development of services. Middleware helps to abstract the distribution and heterogeneity of the underlying computing environment and services available and supports the addition of non-functional values. A number of middleware platforms were developed to add new values for different systems such as enterprise systems [5], cluster computing [1], mobile ad hoc networks [12], and robotics [20]. The mechanisms and approaches are usually based on the reuse of existing methods, protocols, software, and systems to add the needed values, which fits nicely within the general concept of service-oriented architecture (SOA).

As a more recent model, SOC offers a promising way for enabling software vendors to present their software applications as services. It provides a framework to represent business processes as independent modules (services) with clear and accessible interfaces. The interactions among services are done through a standard description language such as XML, which makes it easy to integrate different services to build a business application and address problems related to the integration of heterogeneous applications in a distributed environment. Services can implement a single business process or a set of different processes that are made available for integration with other heterogeneous services. Services can be developed using a wide range of technologies, including SOAP, REST, RPC, DCOM, CORBA and Web Services. The generalized model - SOA - basically involves three main players: the service provider, the service broker and the service requester. The service provider designs and develops a service. The service broker makes this service available to the rest of world through public registries such as Universal Description Discovery and Integration (UDDI) for web services. The service requester locates the entries in the public registry and binds with the service provider to invoke the web services required. There are many researchers and practitioners with many different levels of understanding and use of SOC. According to [21], the general SOA approach makes the following things possible:

- Multiple applications executing in varied platforms can effectively communicate with each other
- Software services are provided to applications through published/discoverable interfaces
- Services are loosely coupled thus reducing dependencies and facilitating reuse
- Service callers are coupled with services needed for the duration of the utilization only
- A relationship is created among participants: service provider, discovery agency, and the service requester
- Interactions between the participants involve the publish, find and bind operations
- Service consumers target specific services through specific interfaces that are exposed by the service registry and discovery mechanisms

In industry, particularly, SOC is becoming more popular and is supported by various large IT organizations. Oracle Corporation provides SOA through their Oracle Fusion Middleware [15], where several SOA based solutions are provided to integrate other tools and applications such as Microsoft products to achieve better interoperability among Oracle-based applications and other applications. Another example is the SAP NetWeaver [25], which provides business process management (BPM) holistic approach to business solutions. NetWeaver BPM provides flexible and efficient models to integrate business processes (as services) to support different application requirements. Progress Open Edge [23] is another example of solutions adapting SOA to provide flexible and effective utilization of services to provide holistic business IT solutions. iWay [24] provides an SOA foundation to enable the integration of business components such as applications, databases, communication transports, and e-business exchanges. In [13], the authors propose a middleware to support the dynamic composition of services in highly heterogeneous pervasive environments.

On the other hand, the research community is also putting a lot of effort to provide usable, efficient and
flexible tools and approaches for SOC. In general several areas are under investigation including web services, SOC, SOM, SOA middleware and integration software. All of which represent variations of the concept of facilitating integration and simplifying application development.

3. Middleware Challenges for WSN

WSN are emerging as a suitable new tool for a spectrum of new applications in recent years. They are easily deployable at a large scale, low power, inexpensive and self-organizing. These unique characteristics make them advantageous over traditional networks. Sensor networks applications were originally motivated by military applications such as target detection, surveillance of enemy activities in a battlefield environment and counterterrorism; however, their many advantages over traditional networks resulted in the development of many other applications that range from infrastructure security to industrial sensing. The design and development of a successful middleware layer for WSN is not trivial. It needs to deal with many challenges dictated by WSN characteristics on one hand and the applications on the other hand [11]:

**Hardware resources:** The advent in microelectronics technology made it possible to design miniaturized low cost devices in the order of cubic centimeters. Limited in energy and individual resources, these tiny devices could be deployed in hundreds or even in thousands in harsh and hostile environments where in some cases no physical contact is possible for replacement or maintenance; therefore, the wireless media is the only way for remote accessibility. Middleware should provide mechanisms for efficient use of the processor and memory while enabling low power communication. A sensor node should accomplish its basic operations (sensing, data processing, and communication) without resource exhaustion. As an example, an energy aware middleware should be able to set most of the device’s components including the radio to turn off most of the time depending on the application needs.

**Scalability and network topology:** the topology of the network is subject to frequent changes due to different factors such as malfunctioning, device failure, moving obstacles, mobility, and interferences. If an application gets bigger, the network should be flexible enough to allow the addition of other nodes anywhere any time without affecting network performance. Also, a middleware should support mechanisms for fault tolerance, self-configuration and self-maintenance of collections of sensor nodes.

**Heterogeneity:** The middleware should provide low level programming models to meet the major challenge of bridging the gap between the hardware technology’s raw potential and the broad needed activities. It should establish system mechanisms interfacing to the various types of hardware and networks, only supported by basic distributed primitive operating system abstractions.

**Network organization:** Unlike traditional networks, sensor networks must deal with resources – energy, bandwidth, and processing power – that are dynamically changing. Another important issue in sensor networks is to support long running applications, thus efficient design of routing protocols is needed so that the network could run as long as possible. An Ad Hoc network resources discovery should be provided, since knowledge of the network is essential for a network to operate properly. A sensor node needs to know its own location in the network in addition to the whole network topology. In some particular cases self-location by GPS is not possible or feasible, or even expensive. Some important issues on system parameters such as network size and density per square mile affect the tradeoffs between latency, reliability, and energy consumption. A middleware should support the robust operation of sensor networks despite these dynamics by adapting to the changing network environment.

**Application knowledge:** another important and unique property of middleware for WSN is dictated by the design principles of application knowledge. However middleware has to include mechanisms for injecting application knowledge in the infrastructure of the WSN. This allows mapping application communication requirements to network parameters that helps fine-tune the network monitoring process.

**Quality of service:** QoS is an overused term with multiple meanings and perspectives from different research and technical communities. In WSN, we can view QoS from two perspectives: application specific and network. The former refers to QoS parameters specific to the application, such as sensor node measurement, deployment, and coverage and number of active sensor nodes. The latter refers to how the supporting communication network can meet application needs while efficiently using network resources such as bandwidth and power consumption. Traditional QoS mechanisms used in wired networks are not adequate for WSN because of constraints such as resource limitations and dynamic topology. Therefore, middleware should provide new mechanisms to maintain QoS over an extended period of time and be able to adapt to changes in QoS requirements and the state of the application. Middleware should be designed based on trade-offs among performance metrics such as throughput, capacity, delivery delay, and energy consumption.

**Security:** WSN are being widely deployed in domains involving sensitive information as in military, healthcare and rescue applications. The large deployment of WSN in such environments increases
their exposure to security risks and malicious attacks. In addition, the wireless medium facilitates eavesdropping and adversarial packet injection to compromise the network's functions, which makes security extremely important. Yet sensor nodes have limited power and processing resources, so standard security mechanisms, which are heavy in weight and resource consumption, are unsuitable. These challenges increase the need to develop comprehensive and efficient security solutions specific to WSN applications to achieve protection, while maintaining desirable performance. Middleware design efforts should concentrate on developing and integrating security in the initial phases of software design.

4. SOM Requirements for WSN

SOM for WSN is necessary to support several, otherwise hard to incorporate, functionalities in the SOC model. These functionalities include not only the functional requirements, but also the non-functional requirements that different services might need to offer. In many application domains, services provide a set of specific functions based on the required application logic which will be different for each service and each application. However, those services and service providers also need to support several common functionalities that are generally irrelevant to the main application. For example most services will have to support service registries and discovery mechanisms and they will also need to provide some level of abstraction to hide the underlying environment and implementation details. Furthermore, all SOC applications need to support some levels of reliability, performance, security and QoS. All of these can be supported and made available through a common middleware platform instead of having to incorporate them into each and every service developed. In general, based on our observations an SOM for WSN should provide the following functionalities:
1. Runtime support for service providers to deploy and advertise services
2. Support for service consumers to discover and use registered services
3. Abstractions to hide the heterogeneity of underlying WSN thru supportive languages and protocols
4. Configurable services
5. Service transparency to client applications
6. Auto-discovery and auto-change mechanisms of service providers
7. Interoperability with a variety of devices and systems
8. Efficient handling of large volumes of data and high communication loads
9. Secure communication and operations
10. Support for QoS requirements

Generally, researchers propose SOM approaches best suited for their underlying environments. Therefore, many of them may provide some of the requirements described above, but most of them do not provide all requirements. Next we discuss some representative approaches.

5. SOM Approaches for WSN

While conducting our research through the different approaches researchers and practitioners are working on, we came across a large pool of examples on SOM for WSN. Several approaches are currently being investigated at the research level while a few are actually being used in practice. The main issue to handle here is the diversity and heterogeneity of the sensors being used and the networks connecting them. Several issues were addressed by this category of approaches like handling complex and/or continuous queries, data access, aggregation and management, service publishing, discovery and deployment, group management, real-time access to data and services, and limited resources. Following is a brief overview of the approaches selected and in Section 6, we discuss some common features and challenges involved.

5.1. SStreaMWare

SStreaMWare [18] is a sensor mediator SOM which provides a query service to access data issued by heterogeneous sensors. The main objective is to provide a hybrid solution focusing on sensor heterogeneity management, scalability and complex query evaluation. The main elements of SStreaMWare are the sensors, the proxies and adapters, the gateways and the control sites. Its architecture consists of three querying services and a discovery/binding service. The Query Service is deployed on the control site and is responsible for query evaluation. The Gateway Service is deployed on every gateway and provides a query service for the sensors managed by that gateway. The Proxy Service is deployed at concerned gateways and provides services to a specific type of sensors. The Lookup Service is for service discovery and binding in addition to managing the service registry.

5.2. USEME

USEME is a service-oriented framework to develop wireless sensor and actor networks (WSAN) applications using middleware support [6]. The main function of the middleware is to handle the publication and discovery of web services, communication related issues, real-time constraints and group management. The architecture of the middleware is divided into five core components including configuration, publication and
discovery, invocation and communication, real-time constraint management and group management. Different parameters such as deadlines and discovery frequency can be changed through its configuration component. The publication and discovery component keeps track of the services available for use. The invocation and communication component takes care of translating the remote commands executed by application programmers into communication packets between remote nodes. The real time constraint management component manages constraints like setting the priorities for activities, specifying maximum execution time for a command and setting time intervals for periodical events. To achieve scalability and efficiency, the middleware groups the nodes that share common behaviors and automatically controls the use of services provided by the groups using the group management component.

5.3. SensorWeb 2.0

In an effort to support the Open Geospatial Consortium’s (OGC) efforts, the Cloud Computing and Distributed Systems (CLOUDS) Lab. at the University of Melbourne (http://www.gridbus.org/sensorweb/), are working on the SensorWeb project (NICTA Open Sensor Web Architecture (NOSA)) [9][16]. In this project the team discusses an SOM to support data collection and manipulation across heterogeneous sensor networks. The main concern is to sustain contiguous access and use of sensor data over extended periods of time. NOSA provides service based access to and management of sensors. It helps integrate sensor networks and distributed computing platforms such as SOA and Grid Computing. The main benefits are: Information processing can be moved from the sensor networks to the backend distributed systems, individual sensor networks can be linked together as services, which can be registered, discovered and accessed by different clients using a uniform protocol. The main components defined for NOSA are composed in four layers: Fabric, Services, Development and Application. It provides sensor services as: (1) Sensor notification, collection and observation; (2) Data collection and aggregation; (3) Sensor co-ordination and data processing; (4) Faulty sensor data correction and management, and; (5) Sensor configuration and directory service.

5.4. OASiS

OASiS [2] is an object-centric, service-oriented, ambient-aware sensor network applications development model. It defines the logical elements necessary to support a wide variety of WSN dataflow applications. It defines a programming framework that facilitates the separation of concerns for application development through a multilayer development process. The core sensor network functionalities are bundled as middleware services including service discovery, service composition, failure detection and node management. Using the OASiS framework, application developers do not require any expertise in sensor network programming. The framework addresses some design challenges for networked embedded sensor systems by supporting dynamic service discovery and configuration, heterogeneity, real-world integration in application design, constraints management, in-network data aggregation and specification of both application-specific and network QoS requirements. The OASiS middleware includes services such as Node Manager, Object Manager, and Dynamic Service Configurator and provides the support to the application services.

5.5. B-VIS

B-VIS (Burapha Vehicle-Infrastructure System) [27] is an SOM targeted to facilitate the utilization of distributed RFID sensor networks for vehicle-based applications. It supports real-time vehicle fleet tracking and control and also supports road traffic monitoring systems. In addition it provides a programming model for developers to retrieve, in real-time, the data collected by the RFID sensor networks. This reduces the complexities of the heterogeneous communication layers and data sources. The B-VIS SOA is based on open methods and loose coupling between the software and hardware components. There are two main layers, the B-Base Gateway layer and the Enterprise Application Server Layer.

5.6. MiSense

MiSense [28] is a service-oriented component-based middleware designed to support distributed sensor applications with various performance requirements. It reduces complexity by imposing a structure on top of the component model in the form of composability restrictions and by offering well-defined service-specific interfaces to the rest of the system. MiSense provides a content-based publish/subscribe service and allows the application designer to adapt the service. The middleware is divided into three layers: The Communication Layer, which provides a content-based publish/subscribe service called MiPSCom that allows the application designer to adapt the service by making orthogonal choices about the communication protocol components for subscription and notification delivery. The Common Services Layer, which provides services common to different applications, such as data aggregation, event detection, topology management and routing. The Domain Layer, which aims to support
domain specific needs, such as data fusion and specific data semantic support to allow the production of application-related information from raw data processing. The programming interface provides functions to allow the user to control and program the WSN as a whole network with different functional characteristics without worrying about the detailed placement of computation and communication.

5.7. SOMDM

The authors propose a service-oriented message driven middleware to support ambient aware WSN [29]. One of the important considerations in WSN is to keep processing overhead at a minimum. Thus the proposed design adopts middleware and SOA advancements to better handle the processing load. This allows the WSN to deal with the real-time demand of query processing, warning systems, seamless integration and execution of process logic. The design also incorporates a data filter for filtering events of interest, which reduces the load on the processing components by using a notification service. This middleware comprises several components: the object code requires nodes of the WSN to follow the ambient programming model; The DataFilter Box separates normal and exceptional events where normal data is directed towards a database and exceptional events are forwarded to a Message Queue; The Messaging Server stores the messages in a queue; The Messaging Components will be notified by the message arrival in the message queue; The DBMS stores the data coming from the DataFilter Box; The Management DashBoard allows the administrator to configure the middleware; The Business Integration Framework allows diverse components, services, legacy systems, messaging servers and clients interfaces to intercommunicate; and The Client Interfaces allow clients to query the sensors and view the warning state.

5.8. (SI)²

Smart Items Services Infrastructure (SI)² [3] is a middleware developed to address the issues of platform-independent service description, deployment and invocation without any knowledge of the underlying smart items network. A smart items network may consist of RFID Systems, WSN, and/or embedded systems. Its architecture is mainly divided into two layers, platform independent and platform dependent. Message Handlers are in the platform dependent layer and their main job is to convert the events generated by the smart items of a specific platform into platform independent events. The Service Lifecycle Managers in the platform dependant layer are responsible for deploying, starting and stopping the services on the network of smart items. The Request Processor, Service Mapper and Service Repository are the main components of the platform independent layer. The Service Mapper is responsible for getting the service description from the middleware and searching for a feasible deployment for it. The Request Processor handles platform-independent requests for a specified service, transmits them to the correct Message Handler and returns the respective results.

5.9. SOA-MM

In [10] an SOA-based middleware for manufacturing systems (SOA-MM) in the form of an integration layer between the shop floor equipment and the enterprise applications is presented. The middleware addresses service granularity, physical distribution and degree of coupling individually. This is done according to the actual capabilities and data flows on the field and control layer irrespective of the type of devices used. The integration layer consists of gateway entities, core services and UDDI registry for publishing core services. A gateway entity is created to map application-specific requests to the core services for a specific device. The core services represent the main engine of the integration layer. Currently there are five core services: device identification, information retrieval (applying application-specific filters to reveal relevant information from device-related information), device positioning, structure analysis (to reveal the structure of complex devices), and event notification.

5.10. ubiSOAP

The main objective of designing ubiSOAP communication middleware [7] is to provide seamless networking of web services that may be deployed on various devices including mobile devices. The architecture of ubiSOAP consists of a multi-radio networking layer and a Web Services (WS) oriented communication layer. The multi-radio networking layer generates and manages Multi-Radio Network Addresses (MRN) assigned uniquely to given applications in a network. This layer also provides the functionality to select the best available network based on the QoS requirements. The WS-oriented communication layer provides point-to-point communication which brings multi-radio multi-network routing to legacy SOAP messaging and group communication. This results in enhancing the SOAP API to meet the corresponding base requirements of pervasive networking environments.

6. Discussion

The job of SOM for WSN is not only to cater for functional properties (such as communication, service creation, discovery and invocation, group management
and real-time operations) but also to support non-functional properties (such as scalability, service reliability, service flexibility, interoperability, and QoS assurance). To achieve this, a middleware should make use of either meta-programming languages or domain-specific languages. Such an initiative has been taken in the case of Service-Oriented Context-Aware middleware which makes use of its own developed domain-specific language to express the context-based rules. The rule management strategy and notification service with the use of domain-specific language can become handy especially in case of service deployment. Since two interactive parties might communicate through multiple paths therefore choosing the best connecting path to serve a given interaction is a key issue to deal with, especially in SOC as this significantly affects QoS [8].

Moreover, since SOM for WSN is based on services of heterogeneous nature, the response time of these services is one of the factors that directly affect its efficiency. To improve efficiency, the response time of the services needs to be improved. Such an improvement is possible by introducing UDP-based SOAP transport instead of HTTP based [17]. However, a typical SOM, which handles large amounts of messages and event notifications coming from numerous services and shares them among different operating components, must have a reliable system in place that ensures an uninterrupted flow of correct and required messages. This can be achieved using different event management techniques.

In addition, SOM performs the job of service orchestration. Separating the application logic from the system logic by a middleware helps in such activity. A middleware based on the concept of Model Driven Architecture (MDA) and SOA is introduced in [26]. SOA design patterns are design techniques that help overcome common obstacles in achieving the strategic goals and benefits associated with SOC [14]. However, development of middleware solutions using SOA design patterns is still at its infancy. Currently, technologies such as web services and P2P are used.

In terms of the requirements identified in Section 4, we found that none of the SOM approaches discussed covers all of them. In fact very few of them address even half of those requirements. Yet, it may be possible to view this as a common practice since most SOM solutions are designed to solve a specific set of problems or to target one type or category of application domains. Therefore, it will be natural to find only those requirements that are most needed in that domain or in those problems. One interesting observation we also came out with is that almost none of the approaches studied address the security issues or propose security as a middleware requirements. This leaves us with an open question about the importance of security in this field. As it is well known, security is important for any type of distributed environments and SOC for WSN is no exception.

In general, we summarize the challenges to address when designing and developing SOM solutions that can satisfy the necessary requirements and achieve their target objectives in the following:

- To be able to maintain an efficient model that can be adapted by others easily.
- To provide the solution in a lightweight implementation and support low resource devices.
- To address access control and security to avoid problems especially in mission-critical applications and sensitive operating environments.
- To conform to current (and future) standards that would make it easy to adapt the solution and make it usable across a larger community.

7. Conclusion

In this paper we discussed the general requirements of SOM for WSN and reviewed some representative approaches in the field. The approaches used carry some similarities in the sense that all of them agree on a common goal: to effectively create, publish, discover and use services. However, the approaches used were also significantly different in how they provide this. In addition, the target benefits of the approaches differ where targets like interoperability, QoS support, real-time support, heterogeneity support, scalability, and context-awareness are involved. We discussed a number of SOM for WSN projects and highlighted their main objectives and approaches. Table 1 provides a summary of these projects. SOM is generally destined to handle the publication and discovery of services, communication, and efficient and reliable event management among different services. In addition, SOM solutions need to support efficient handling of the heterogeneous resources and functionalities of the WSN applications. Moreover, due to the need to exchange large volumes of data, it is necessary to have a reliable and efficient system in place to ensure an uninterrupted flow of correct and required messages. It is also necessary to maintain acceptable levels of control, security and reliability through the SOM. Finally, to further enhance the approach, we must have dynamic, adaptive and auto-configurable SOM architectures for effective operation for WSN applications.
Table 1. Summary of the SOM approaches and their main features.

<table>
<thead>
<tr>
<th>SOM</th>
<th>Platform/ Environment</th>
<th>Main Feature</th>
<th>Other features</th>
<th>Requirements (explicitly covered)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSStreaMWare</td>
<td>Sensor networks (wired and wireless)</td>
<td>Support for query-based data access</td>
<td>Heterogeneity management, scalability and complex queries handling</td>
<td>3, 4, 6, 7 and 8</td>
</tr>
<tr>
<td>USEME</td>
<td>Wireless sensor and actor networks</td>
<td>Handles publication and discovery of web services</td>
<td>Supports group management and real-time constraints</td>
<td>1, 2, 4, 6 and 8</td>
</tr>
<tr>
<td>SensorWeb 2.0</td>
<td>Sensor networks (wired and wireless)</td>
<td>Supports data collection and manipulations across heterogeneous sensors</td>
<td>Open source architecture, sustains contiguous access to sensor data</td>
<td>3, 4, 7 and 8</td>
</tr>
<tr>
<td>OASiS</td>
<td>WSN</td>
<td>Development environment based on separation of concerns</td>
<td>Supports QoS, failure detection, node management and dynamic service</td>
<td>1, 2, 3, 5, 8 and 10</td>
</tr>
<tr>
<td>B-VIS</td>
<td>Distributed RFID and Sensor networks</td>
<td>Real-time tracking and monitoring through sensors</td>
<td>Provides a programming model to retrieve real-time data and integrates</td>
<td>3, 7 and 8</td>
</tr>
<tr>
<td>MiSense</td>
<td>WSN</td>
<td>Content-based publish/subscribe model</td>
<td>Provides programming API, domain specific support, and flexible data management support</td>
<td>1, 2, 3, 7 and 8</td>
</tr>
<tr>
<td>SOMDM</td>
<td>WSN</td>
<td>Multi-component architecture to reduce data processing loads</td>
<td>Data filtering mechanism, notification mechanism and DBMS support</td>
<td>3, 5, 7 and 8</td>
</tr>
<tr>
<td>(Si)</td>
<td>Smart items network</td>
<td>Platform independent service description, deployment and invocation</td>
<td>Support for wide range of devices including sensor, RFID, and embedded devices</td>
<td>1, 2, 3 and 7</td>
</tr>
<tr>
<td>SOA-MM</td>
<td>Enterprise systems and shop floor equipment</td>
<td>Integrating shop floor operations with enterprise applications</td>
<td>Support for heterogeneous devices and networks and for loosely coupled services</td>
<td>3, 4, 7 and 8</td>
</tr>
<tr>
<td>ubiSOAP</td>
<td>General distributed environments (wired, wireless and mobile)</td>
<td>Seamless networking of web services</td>
<td>Multi-radio networking and selection of best network for QoS requirements</td>
<td>2, 4, 5, 7 and 10</td>
</tr>
</tbody>
</table>

Requirements:
1. Runtime support for service providers to deploy and advertise services
2. Support for service consumers to discover and use registered services
3. Abstractions to hide the heterogeneity of underlying environments thru supportive languages and protocols
4. Configurable services
5. Service transparency to client applications
6. Auto-discovery and auto-change mechanisms of service providers
7. Interoperability with a variety of devices and systems
8. Efficient handling of large volumes of data and high communication loads
9. Secure communication and operations
10. Support for QoS requirements

References


[23] Progress Open Edge, Supporting Business and Technology Change with Progress, white paper, Progress Software corp., USA, 2007, URL: http://www.progress.com/progres_software/docs/schenker_final.pdf


