A Service-Oriented Component-Based Middleware Architecture for Wireless Sensor Networks

Kavi Kumar Khedo† **and R.K. Subramanian**††,

Faculty of Engineering, University of Mauritius, Reduit, Muairitus

Summary
The use of sensor networks in different spheres of the modern society is emerging as a new trend. However, the integration and coordination of heterogeneous sensors is still a challenge, especially when the target application scenario is susceptible to constant changes. To enable a wider adoption of sensor network technologies, we must address a variety of constraints inherent in sensor network operation and provide a significantly rich level of abstraction to application users supported by efficient and robust optimization techniques. In this paper, we propose, MiSense, a service-oriented component-based middleware layer in order to support distributed sensor applications with various performance requirements. MiSense 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 breaks up the middleware design into fine, self-contained and richly interacting components in order to resolve the tension between the optimization requirements for specific scenarios and the need for flexibility and reusability for developing energy efficient wireless sensor networks applications.

Key words:
Sensor networks, Middleware, Service-oriented approach, Communication model, Cluster-based routing.

1. Introduction

Wireless sensor networks have the potential to be tremendously beneficial to the modern society. Currently, wireless sensor networks are beginning to be deployed at an accelerated pace [1]. Embedded sensing will enable new scientific exploration, lead to better engineering, improve productivity, and enhance security. Research in sensor networks has made dramatic progress in the past decade, bringing these possibilities closer to reality [2]. However, the integration and coordination of heterogeneous sensors is still a challenge, especially when the target application scenario is susceptible to constant changes. Such systems must adapt themselves in order to fulfill requirements that can also change during the system runtime. Moreover, the changes that occur in such scenarios require services located at different places during the system runtime. Due to the dynamicity of this context, system adaptations must take place very quickly, requiring that decisions for adaptation are taken autonomously by the system without waiting for human operator's directions.

Any design for sensor networks is subject to tight constraints in terms of energy, processing power and memory. These constraints frequently drive developers to pursue vertically integrated solutions that are highly-optimized for specific scenarios [13]. Literature in this area presents a wide range of protocols and subsystems [15] that make widely differing assumptions about the rest of the system and how its parts should interact. The extent to which these parts can be combined to build usable systems is quite limited.

In order to produce running systems, research groups have produced vertically integrated designs in which their own set of components are specifically designed to work together, but are unable to interoperate with the work of others. This inherent incompatibility greatly reduces the synergy possible between research efforts and impedes progress. Thus, current solutions highly optimized for specific scenarios but lack flexibility. Moreover, there is tension between the need for flexibility and the efficiency costs of abstractions. It can therefore be concluded that the factor currently limiting research progress in sensor networks today is not any specific technical challenge (though many remain, and deserve much further study) but is instead the lack of an overall sensor network architecture.

In this paper, we are proposing, MiSense, a component-based service-oriented middleware architecture with a set of generic services that provides an abstraction layer between applications and the underlying network infrastructure. We have identified the essential services and their conceptual relationships for an overall sensor network architecture. Such a decomposition would make it possible to compose components in a manner that promotes interoperability, transcends generations of technology, and allows innovation. The middleware architecture promotes a content-based publish/subscribe communication model and proposes dynamic reconfiguration through reflective methods.

Manuscript received March 5, 2009
Manuscript revised March 20, 2009

2. Related Works

Middleware development in the growing and promising field of sensor networks is a major challenge in order to facilitate the programmer task and bridge the gap between the applications and the hardware. However, most of the current projects [4, 7, 8, 9, 11, 17] on sensor middleware are at an early stage, focusing on developing algorithms and components for data aggregation, localization, service discovery, routing, and synchronization. These projects, however, often lack attention for integrating these algorithms and components into a generic middleware architecture, and for helping application developers to compose a system that exactly matches their requirements. Consequently, developing and deploying end-to-end applications for sensor networks in a realistic business context remains highly complex.

Traditional distributed programming abstractions like Remote Procedure Calls (RPC), or the Distributed Object Model (DOM) have traditionally simplified and enabled the implementation of complex distributed systems. Unfortunately, these abstractions and middleware architectures cannot be simply applied to sensor networks due to the new characteristics and peculiarities of the latter. Existing approaches have to be revisited or new approaches have to be developed to meet the requirement of sensor networks. Research into middleware and programming environments has become a more important issue recently as researchers have realized that sensor networks are difficult to use, program, and manage [6, 12, 14, 16]. It is argued that this difficulty has artificially impeded the adoption of the technology outside of the computer science community. In response to this, a wide variety of different systems that make different assumptions and tradeoffs have been proposed. These systems range from very low level mechanisms to high level concepts that abstract the notion of programming.

Moreover, sensor network applications are becoming more complex due to the use of different kinds of mobile and sophisticated sensors, which provide advanced functionalities and are deployed in dynamic scenarios where context-awareness is needed. To support those emerging applications, an adaptable underlying infrastructure is necessary. Current state-of-the-art middleware for sensor networks present important non-negligible drawbacks that make them useless in the context of such new emerging applications, because: (i) the assumption that the network is composed only by a homogeneous set of basic or very constrained low-end sensors; (ii) the lack of intelligence in such network compromises the adaptability required to deal with changing operation conditions, e.g. lack of QoS management and control [18].

According to its particular assumptions, each of the proposed middleware solutions for wireless sensor networks draws on selected aspects of traditional middleware for distributed systems, such as distributed databases or publish/subscribe systems. Most solutions fit into one of the following categories:

- database-inspired approaches, which use SQL-like queries;
- tuple space approaches, which build on the tuple space abstraction made popular by Linda [22];
- event-based approaches, which use event correlation to aggregate sensor data; and
- service discovery based approaches, which use service discovery protocols to locate sensors that can meet applications' data requirements.

SINA (System Information Networking Architecture) [5] models the network as massively distributed objects. SINA is cluster-based middleware, and its kernel is based on a spreadsheet database for querying and monitoring. Each logical datasheet comprises of cells, and each cell represents a sensor node attribute (in the form of a single value, such as power level and location, or multiple values, such as temperature changes history). Each cell is unique, and each sensor node maintains the whole datasheet. The sensor network as whole is a collection of datasheets. The spreadsheet approach is the abstraction that allows information management to meet application changes and needs.

Besides cluster based middleware, much research has focused on query based systems. Systems such as TinyDB [8] and Cougar [19] view the sensor network as an online, distributed database. Instead of explicitly programming nodes on the network, users simply access data by using a declarative query language similar to SQL. The query is then propagated to the relevant nodes identified by the query and a reply is sent back to the user. Since SQL provides support for simple reduction functions such as average, minimum, and maximum, such systems are able to efficiently aggregate data by employing a spanning tree routing structure.

Another class of middleware approaches is inspired by mobile code and mobile agents [20]. There, the sensor network is tasked by injecting a program into the sensor network. This program can collect local sensor data, can statefully migrate or copy itself to other nodes, and can communicate with such remote copies. SensorWare [21] is a middleware implementation of this class. Yet another approach to sensor network middleware is based on the notion of events. There, the application specifies interest in certain state changes of the real world. Upon detecting such an event, a sensor node sends a so-called event notification towards interested applications. The application can also specify certain patterns of events, such that the application is only notified if occurred events match this pattern. DSWare [9] is a representative of this class of middleware. DsWare is a database-like abstraction approach tailored to sensor networks on the basis of event detection.

Mires [10] proposes an adaptation of a message-oriented middleware for traditional fixed distributed systems. Mires provides an asynchronous communication model that is suitable for WSN applications, which are event driven in most cases, and has more advantages over the traditional request-reply model. It adopts a component-based programming model using active messages to implement its publish-subscribe-based communication infrastructure. Maté [3] is an architecture for constructing application specific virtual machines that executes on top TinyOS. Using this architecture, developers can easily change instruction sets, execution events, and virtual machine subsystems. Maté provides a simple programming interface to sensor nodes. For example, a sense-and-send program can be written with six instructions.

Another middleware, Impala [4] designed for use in the ZebraNet project, considers the application itself exploiting mobile code techniques to change the functionality of the middleware executing at a remote sensor. The key to energy efficiency for Impala is for the sensor node applications to be as modular as possible, enabling small updates that require little power during transmission. Unlike Impala and Maté, MiLAN (Middleware Linking Applications and Networks) [7] has an architecture that reaches the network protocol. MiLAN is intended to sit on top of multiple physical networks. It acts as a layer that allows network-specific plug-in to convert MiLAN commands to protocol-specific ones that are passed through the usual network protocol stack. Therefore, MiLAN can continuously adapt to the specific features of whichever network is being used in the communication. MiLAN uses graph theory and presents a mechanism to select the best nodes in a sensor network.

In this section, we presented concrete middleware approaches for sensor networks with different underlying programming paradigms (e.g., database approach, agent-based approach, event-based approach). These paradigms are not new, but require significant adaptation for use in sensor networks. The approaches differ with respect to ease of use, expressiveness, scalability, and overhead. Most of the projects we have mentioned are at an early stage, focusing on developing algorithms and components of WSN middleware. One primordial issue is to provide energy-efficiency requirements while providing a high-level abstraction that addresses sensor node heterogeneity. Another crucial challenge is developing an easy-to-use, expressive programming interface while meeting different sensor network application challenges.

3. MiSense: A Service-Oriented Component-Based Middleware Layer

MiSense promotes a service-oriented middleware component framework that can reduce complexity by imposing 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 aims at fixing the service interface at a level of abstraction that will maximize the gains in productivity, while keeping those parts of the architecture with significant impact on the performance flexible enough to be able to benefit from domain-specific optimization. MiSense provides a well-defined content-based publish/subscribe service, but allows the application designer to adapt the service by making orthogonal choices about the communication components for subscription and notification delivery, the supported data attributes, and a set of service extension components.

The middleware is divided in three parts or layers indicating that they are partly using each other in a specific order. Figure 1 below presents an overview of the layers of the proposed middleware, and a description of each layer is provided.

The bottom layer of the MiSense middleware is called the Communication Layer. It provides a well-defined content-based publish/subscribe service, MiPSCom, that allows the application designer to adapt the service by making orthogonal choices about the communication protocol components for subscription and notification delivery. A major design goal of the content-based publish/subscribe communication model is to separate out those service sub-tasks which are expected to have large impact on the resource usage. This decomposition strives to give an application designer a simple and flexible means to select protocol components and data attributes according to his needs, and to give him more fine-grained control over the publish/subscribe service through the concept of extension components.

The resource management layer coordinates the resource sharing based on application needs passed through the upper layers. Services provided by upper layers may need some resource sharing support, which is encapsulated in the communication layer. As an application uses such a service, the corresponding layer asks for the communication layer to manage the access control to the required resources. Indeed, the resource management layer commands the allocation and adaptation of resources, such that the QoS requirements specified by the applications can be met. Resource allocation focuses on generating an initial solution when the cluster is formed, while resource adaptation controls the runtime behaviour of the cluster. Both of these steps need to solve the problem of determining the scheduling of applications onto corresponding resources and the adjustment of system knobs. The primary responsibility of the RML is to provide the means for registering sensor and actuator networks to the middleware and tracking their resource usage levels, (i.e., residual device energy and available network bandwidth).

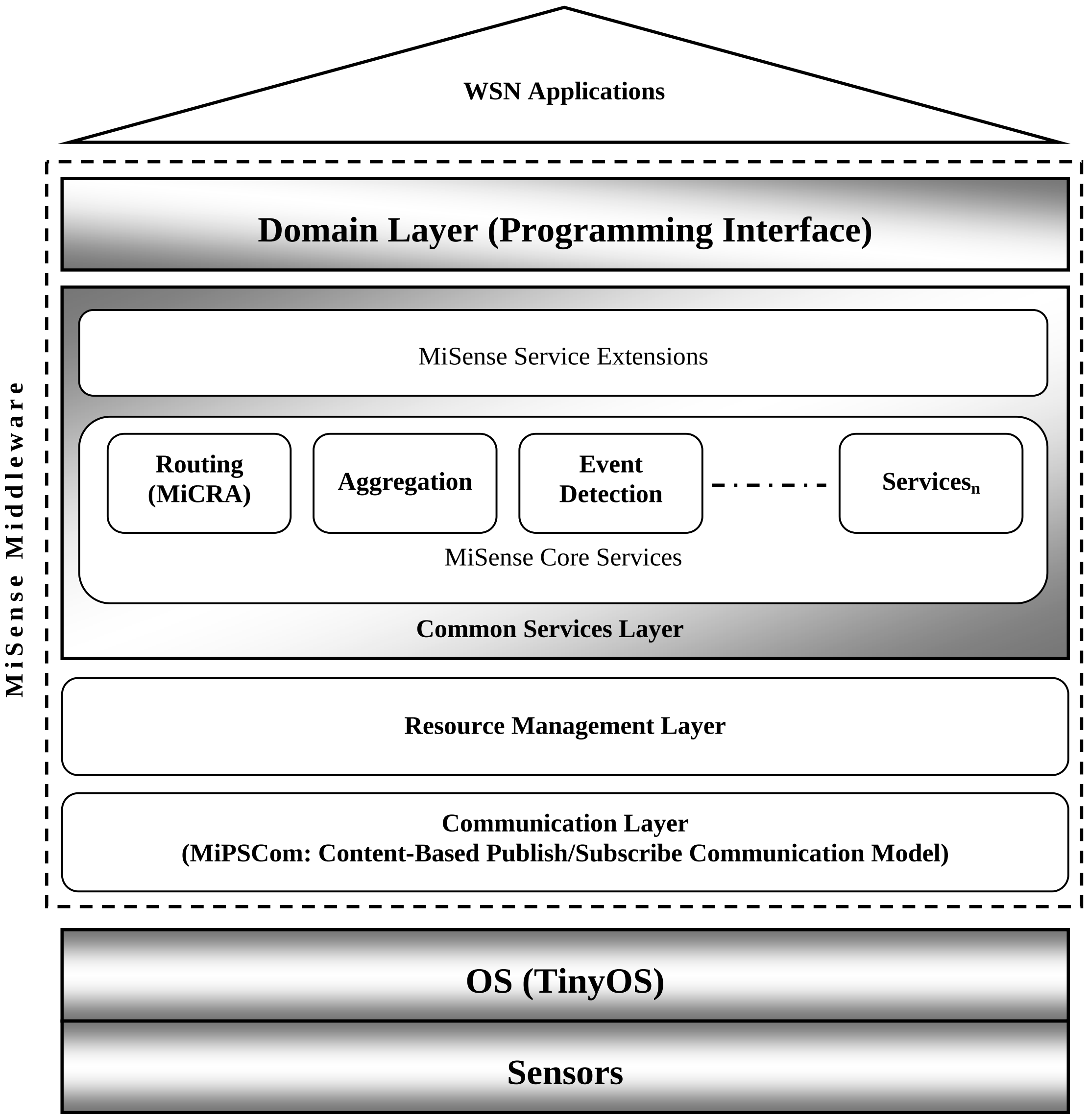
WSN Applications
Domain Layer (Programming Interface)
MiSense Service Extensions
Routing (MiCRA)
Aggregation
Event Detection
Services$_n$
MiSense Core Services
Common Services Layer
Resource Management Layer
Communication Layer (MiPSCom: Content-Based Publish/Subscribe Communication Model)
MiSense Middleware
OS (TinyOS)
Sensors

Fig. 1 The MiSense overall architecture

The intermediate layer is called Common Services Layer, which provides services that are common to different kinds of applications, such as data aggregation, event detection, topology management and routing. The set of generic services provided by the MiSense middleware offers flexibility in the design of WSN applications since it provides accepted standards for representing and packaging data, describing the functionality of services, and facilitating the search for available services which can be invoked to meet application requirements [5]. The MiSense middleware services are capable of maintaining acceptable performance levels as the network grows. Sensor network topology is subject to frequent changes owing to factors such as malfunctioning, device failure, moving obstacles, mobility, and interference. The MiSense middleware aims at supporting sensor networks' robust operation despite these dynamics by adapting to the changing network environment. Refined and optimised services can be plugged into the framework through the MiSense Service Extensions component without modification of existing code.

The top layer is called Domain Layer and has the goal to support domain specific needs, such as data fusion support and specific data semantic support to allow the production of application-related information from raw data processing. Fuzzy classifiers, special kinds of mathematical filters and functions that can be reused by applications of the same domain will be found in this layer. The programming interface provides a set of functions that will allow the user to control and program the sensor network as a whole network with different functional characteristics without worrying about the detailed placement of computation and communication. This style of programming will allow the programmer to be more productive and will allow unique optimizations to be made to prolong the lifetime of the sensor network application.

4. MiSense Services

We have developed a suite of middleware services which support the features of our architecture. The middleware provides a layer of network abstraction, shielding the application developer from the low-level complexities of sensor network operation such as resource management and communication. It gracefully handles the decomposition of desired application behaviour to produce node-level executable code for an object-centric, service-oriented WSN application.

The proposed middleware has been designed using a service-oriented approach [23]. For an external point of view, applications are service requestors and sink nodes are service providers. Sink nodes release the descriptions of the services provided by the WSN and offer access to these services. From an internal point of view, sinks are the service requestors and sensor nodes are the service providers. Sensors send the descriptions of their services to sink nodes, which keep a repository of the service descriptors of each type of existing sensor in the network.

The generic middleware services in MiSense include data aggregation, event detection, and topology management. We also propose a hierarchical cluster-based routing scheme named MiCRA, which is suitable for different types of sensor networks applications such as habitat and environmental monitoring applications. The proposed routing scheme is based on the fact that the energy consumed to send a message to a distant node is far greater than the energy needed for a short range transmission. The main aim of MiCRA is to efficiently maintain the energy consumption of sensor nodes by involving them in multi-hop communication within a particular cluster and by performing data aggregation and fusion in order to decrease the number of transmitted messages to the sink. MiCRA uses two important parameters in order to prolong the lifetime of the sensor network. The first parameter is the “residual energy” of nodes which is used to probabilistically select an initial set of cluster heads and the second one is the intra-cluster “communication cost” which is used to break “ties”. A tie in this context means that a node falls within the “range” of more than one cluster head, including the situation when two tentative cluster heads fall within the same range. MiCRA consists of electing 2 levels of cluster-heads (CHs). The first level election uses the same CHprob equation as in the HEED algorithm [24], whereas the second level election is different from the first one where only the first level CH participate and their CHprob is calculated according to the following equation:

$$CH_{\Pr ob} = \left(\frac{Eresidual}{E\max} \times \left(1 - \frac{ClusterSize}{NumNodes} \right) \right)$$

In the 2nd level CH election, the 2nd level CHs have an unequal topology, where the 2nd level CHs which are near the base station have less members associated with it compared to those that are far away. The advantage derived from such topology is that it prevents second level cluster heads from depleting fast due to heavy relay and intra cluster traffic. In such case, a 1st level CH will join the 2nd level CH with highest residual energy. To achieve such a topology, each node decreases its competition radius as it nears the BS hence resulting in an unequal topology. The main objective of MiCRA is that it is more efficient for the relaying of packets to the base station. In this new scheme, fewer nodes are involved for transmitting packets to the base station compared to HEED thus reducing the overall consumption of energy in the network and thus helping in prolonging the network lifetime.

The competition radius (R_{comp}) is a function of a node distance to the base station is given by:

$$R_{comp} = \left(1 - c\,\frac{d_{\max} - d(s_i, BS)}{d_{\max} - d_{\min}} \right) R^0_{comp}$$

R^0_{comp} is the maximum competition radius which is predefined.

d_{max} and d_{min} denote the maximum and minimum distance between sensor nodes and the base station.

$d(s_i, BS)$ is the distance between a node s_i and the base station.

c is a constant coefficient between 0 and 1.

MiCRA Algorithm Design

I. Initialise

(a) Calculate communication range of node using formula (6):

$$R_{comp} = \left(1 - c\frac{d_{\max} - d(s_i, BS)}{d_{\max} - d_{\min}}\right) R^0_{comp}$$

(b) For each node within communication range
Add node id of each neighbour found in an array (Snbr)
(c) <u>Calculate cost of each node based on residual energy of node</u>
(d) For each neighbour found in Snbr array
Send cost
(e) Calculate cluster head probability based on formula (2)

$$CH_{\Pr ob} = \left(\frac{Eresidual}{E\max} \times \left(1 - \frac{ClusterSize}{NumNodes}\right)\right)$$

(f) Set "Is_Final_CH" attribute to False

5. MiSense Communication Model

The communication layer is based on an enhanced publish/subscribe scheme which has been named as the MiPSCom, MiSense Content-based Publish/Subscribe Communication Model. The core decomposition of the proposed communication model is discussed in this section. Table 1 shows the enhanced publish/subscribe scheme.

Table 1. The enhanced publish/subscribe API that is provided by communication model. A square bracket represents a set of constraints (C), metadata (M) or attributevalue pairs (A).

Enhanced Publish/Subscribe API	

Subscriber:	Subscribe([C] [M])
	Unsubscribe()
	Notify([A] [M])

Publisher:	Publish([A] [M] , push)
	Listener([C] [M])

Matching:	Matching([C] , [A])

Figure 2 shows the decomposition of the communication model. The Publish/Subscribe service is distributed and the figure represents an instance of the model on one sensor node. A publish/subscribe application is divided into a variable number of Publisher and Subscriber components. A Publisher component can listen for subscriptions, collect data and publish notifications and Subscriber components can issue subscriptions and receive matching notifications. The Broker component provides the publish/subscribe service to the application, it manages the subscription table and it can apply the matching algorithm to filter out notifications that do not match a registered subscription.

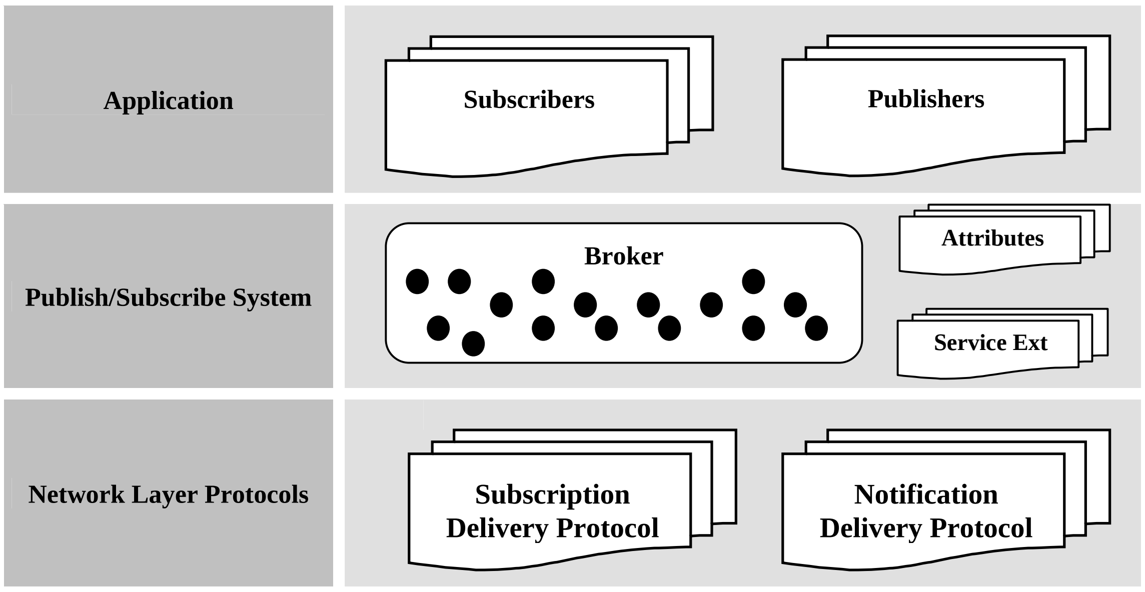

Fig. 2 The MiPSCom Architecture.

The data (“events") that subscribers can subscribe to and publishers can publish are encapsulated in Attribute components. In addition to a data collection interface, an Attribute component must provide a matching interface that compares two of its data items based on an attribute-specific operator. The motivation is twofold: First, an Attribute component represents functionality that Publisher components should be able to reuse and access independent of the specific attribute properties (data type, metric, etc.). Secondly, matching operators are usually attribute dependent: for example, when sensor readings are affected by hardware-related jitter, the operator “=" should not be interpreted as the exact equality of two values. To increase modularity and keep the core matching algorithm decoupled, this information should be provided by the particular Attribute component. Within the network, all attributes and operators are represented by integral identifiers. Attribute identifiers are globally unique, while operator identifiers are unique within the scope of a particular attribute. The AttributeCollector component structures access to the attributes: it maps a request based on the attribute/operator identifier to an actual Attribute component that is registered at compile time (but could even be added at runtime by dynamic over the air code updates).

In MiPSCom, the proposed communication model, the publisher publishes its interface (Listener), including the events it will notify. A subscriber registers interest in events indicating, where appropriate, constraints on the event parameters. The publisher notifies the subscriber of event occurrences that match the subscriber's registration. The broker service acts as a mediator between the publisher and the subscriber decoupling the subscriber and the publisher in space, flow and time, undertaking event filtering and event storage and, at the same time, providing services such as message buffering and message forwarding to disconnected subscribers. In MiPSCom subscribers register their interest in events by typically calling a Subscribe() operation on the event service without knowing the publishers of these events. A symmetric operation Unsubscribe() terminates a subscription. To generate an event, a publisher calls a Notify() operation on the event service. The event service directs the call to all relevant subscribers so that every subscriber receives a notification for every event conforming to its registration.

The key elements in the proposed communication model are the notification service and the buffer where the messages are queued before they are passed to subscribers. The notification service takes responsibility to inform the subscribers when a new message arrives. In this way, it allows the asynchronous communication as producers and consumers are fully decoupled. This loose coupling is the prime advantage of this kind of communication in the context of ad-hoc and pervasive environments such as wireless sensor networks.

6. Proposed Adaptation Strategy

Distributed sensor applications demand a high degree of flexibility and adaptability in order to deal with dynamic changes in application requirements and sensor environments. They can benefit greatly from knowing the status inside the underlying layers, and in the computational and physical environment. Therefore, we introduce the notion of computational reflection to the MiSense sensing architecture, bringing network and system monitoring support to the level of sensor applications. Computational reflection [25, 26] is a technique that allows a system to observe and maintain information about itself (meta-data) and use this information to change its behavior (adapt). In other words, the system maintains a causally-connected self representation. This is achieved by processing at two well-defined levels: functional level (also known as base or application level) and management (or meta) level. An important part of the reflection is the reification process - the capture and observation of the base level states.

The middleware inspection capacity allows an application to request information on the current execution context. The request and the respective response are represented as SOAP messages. From the analysis of the provided information, the application may decide to modify the system behavior, changing some previously registered QoS parameter or execution policy. The adaptation module keeps a table to register the parameters that each application requests to monitor. Monitoring components existent in the sensor nodes periodically check the values of requested parameters.

Adaptation policies are pre-registered in the system as sets of actions to be performed when the application QoS requirements are not being fulfilled for a given execution context. Adaptation policies created for the proposed middleware are: (i) increase the data reliability (data accuracy); (ii) decrease the energy consumption; (iii) increase the available bandwidth. A policy of decreasing the energy consumption may be implemented by two actions: decreasing the data rate and turning off some sensors.

7. Conclusion

The MiSense middleware provides an abstraction layer between applications and the underlying network infrastructure. Besides supplying an abstract programming model to WSN applications, it keeps the balance between application QoS requirements and the network lifetime.

The middleware is in charge of decisions about communication protocols, network topologic organization, sensor operation modes and other infrastructure functions typical of WSNs. The middleware monitors network and application execution states performing a network adaptation whenever it is needed, with or without application interference. A major design goal of the presented communication model is to separate out those service sub-tasks which are expected to have large impact on the resource usage. This decomposition strives to give an application designer a simple and flexible means to select protocol components and data attributes according to his needs.

The main contributions of the proposed middleware are three-folded. First, MiSense 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. Second, the services provided by the middleware are accessed in a flexible way through a standard and high-level interface. MiSense breaks up the middleware design into fine, self-contained and richly interacting components in order to resolve the tension between the optimization requirements for specific scenarios and the need for flexibility and reusability for developing energy efficient wireless sensor networks applications. Finally, the provided services of decision about network configuration and of dynamic adaptation aim to increase the network global lifetime, while meeting the applications requirements.

We believe that MiSense is well suited for sensor networks, in order to satisfy the resource constraints and it can enable a wealth of new sensor based services. MiSense provides a strong programming abstraction that simplifies application development while still maintaining flexibility. Our work has attempted to provide a simpler, more productive interface to the sensor network. By approaching the problem from the architectural level, common, low level functions were factored out and provided as middleware services. The resulting architecture of MiSense is sufficiently general to support the properties identified as necessary to adequately fulfill the middleware role.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Sensor networks: A Survey,” *Computer Networks*, vol. 38, no. 4, pp. 393-422, March 2002.

[2] D. Culler, D. Estrin, and M. Srivastava, "Overview of Sensor Networks," http://doi.ieeecomputersociety.org/10.1109/MC.2004.93, *Computer*, vol. 37, no. 8, 2004, pp. 41-49.

[3] P. Levis and D. Culler, "Mate: A Tiny Virtual Machine for Sensor Networks," *Proc. 10th Int'l Conf. Architectural Support for Programming Languages and Operating Systems* (ASPLOSX), ACM Press, 2002, pp. 85-95.

[4] T. Liu and M. Martonosi, "Impala: A Middleware System for Managing Autonomic, Parallel Sensor Systems," *Proc. ACM SIGPLAN Symp. Principles and Practice of Parallel Programming* (PPoPP 03), 2003, pp. 107-118.

[5] C. Shen, C. Srisathapornphat, and C. Jaikaeo, “Sensor Information Networking Architecture and Applications”, *IEEE Personal Communications*, August 2001, pp. 52–59.

[6] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy efficient communication protocol for wireless microsensor networks. In *Proceedings of the Hawaii International Conference on System Sciences*, January 2000.

[7] A. Murphy and W. Heinzelman, “MiLan: Middleware linking applications and networks,” University of Rochester, Tech. Rep. TR-795, 2002.

[8] S.R. Madden, M.J. Franklin and J.M. Hellerstein, "TinyDB: An Acquisitional Query Processing System for Sensor Networks," *ACM Trans. Database Systems*, 30(1), 2005.

[9] S. Li, S. Son, and J. Stankovic, "Event Detection Services Using Data Service Middleware in Distributed Sensor Networks," *Proc. 2nd Int'l Workshop Information Processing in Sensor Networks* (IPSN 03), LNCS 2634, Springer, 2003, pp. 502-517.

[10] E. Souto, et al., "A Message-Oriented Middleware for Sensor Networks," *Proc. 2nd Int'l Workshop Middleware for Pervasive and Ad-Hoc Computing* (MPAC 04), ACM Press, 2004, pp. 127-134.

[11] L. St. Ville and P. Dickman, "Garnet: A Middleware Architecture for Distributing Data Streams Originating in Wireless Sensor Networks," *Proc. 23rd Int'l Conf. Distributed Computing Systems Workshops* (ICDCSW 03), IEEE CS Press, 2003, pp. 235-241.

[12] K. Romer, “Programming paradigms and middleware for sensor networks,” *GI/ITG Workshop on Sensor Networks*, Germany, 2004, pp. 49-54.

[13] D. Chen and P.K. Varshney, “QoS support in wireless sensor networks: a survey,” *Proc. of Int. Con. On Wireless Networks (ICWN)*, Las Vegas, 2004.

[14] Y. Yu, B. Krishnamachari, and V. E. Prasanna, “Issues in designing middleware for wireless sensor networks”, *IEEE Network Magazine*, vol. 18, 2004, pp.15-21.

[15]J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, “System Architecture Directions for Networked Sensors,” *In ASPLOS 2000*, Cambridge, USA, Nov. 2000.

[16] W.B. Heinzelman, A.L. Murphy, H.S. Carvalho, and M.A. Perillo, “Middleware to Support Sensor Network Applications”, *IEEE Network*, January 2004, pp. 6–14.

[17] K. Aberer, M. Hauswirth, and A. Salehi, “The Global Sensor Networks middleware for efficient and flexible deployment and interconnection of sensor networks,” Tech. Rep. LSIR-REPORT-2006-006, Ecole Polytechnique F´ed´erale de Lausanne, 2006.

[18] K. Römer, O. Kasten, and F. Mattern, “Middleware challenges for wireless sensor networks,” *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 6, no. 4, 2002.

[19] Y. Yao and J. Gehrke, “The Cougar Approach to In-Network Query Processing in Sensor Networks,” *In SIGMOD*, 2002.

[20] T. Umezawa, I. Satoh, and Y. Anzai, “A mobile agent-based framework for configurable sensor networks,” *Proc. of 4th Int. Workshop on Mobile Agents for Telecom Applications*, 2002, pp. 128-140.
[21] A. Boulis, C. C. Han, and M. B. Srivastava, “Design and Implementation of a Framework for Programmable and Efficient Sensor Networks,” *In Proceedings of the 1st ACM/USENIX Conference on Mobile Systems,* Applications, and Services, pp. 187- 200, ACM Press, New York, May 2003.
[22] D. Gelernter, “Generative communication in Linda”, ACM Computing Surveys, 7(1), 1985, 80–112.
[23] Delicato, F. et al., “Service Oriented Middleware for Wireless Sensor Networks”. TR. NCE04/04. Available in http://www.nce.ufrj.br/labnet/research/sensornet/publications.htm, 2004.
[24] O. Younis and S. Fahmy, "Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach," In Proceedings of IEEE INFOCOM, vol. 1, March 2004.
[25] P. Maes, “Concepts and experiments in computational reflection”, In Proceedings of OOPSLA, 1987.
[26] B. Smith, “Reflection and Semantics in a Procedural Language”, PhD thesis, Massachusetts Institute of Technology, January 1982.

Kavi Kumar Khedo is currently a senior lecturer and Head of the Department of Computer Science and Engineering at the University of Mauritius, Reduit, Mauritius. He is an active member of the Context-Awareness Research Group and Mobile and Ubiquitous Computing Research Group at the University of Mauritius. Kavi Khedo has acted as reviewer for a number of international conferences including CSITEd 2007, ICIC 2007, WCSN 2007 and InSITE 2008. He is currently working on energy-efficient middleware technologies for wireless sensor networks. His research interests are also directed toward mobile ad-hoc networks, context-awareness, mobile and ubiquitous computing.

R. K. Subramanian is currently a Professor with the Department of computer Science and Engineering, Faculty of Engineering, University of Mauritius. He has over 40 years of teaching and research experience. He received his Ph.D Degree . in Computer Science from the Indian Institute of Technology, Delhi. His research interests include Mobile and Ubiquitous Computing, Biometric Based Security, Knowledge Modeling and Distributed systems.