# Service Provisioning Middleware for Wireless Sensor Network

**S. Sasirekha and S. Swamynathan**

**Abstract** Earlier, wireless sensor network (WSN) applications tended to follow the traditional format of being specific to a device. Later, when applications evolved integrating heterogeneous devices, it rendered difficulty in enforcing a common standard among all the diverse devices. In order to handle this, a lot of WSN middleware platforms emerged to bind the application interaction with the heterogeneous devices through heterogeneous interfaces. This started increasing the service-based applications while decreasing the device-based applications. Apart from not only providing the classic task of reading the information from the sensor network, the middleware support were extended to address interoperability, management, security, and privacy. However, still there exists an important issue, which many of the existing middleware fail to address. For instance, the network design scenario varies depending on the application context. However, most of the existing middleware operate on the default network infrastructure and data dissemination protocol to collect the data and perform other tasks on the network. Therefore, there is a requirement to include support for customizing the network configuration for an application requirement with respect to its context. Hence, in this work, a service provisioning middleware based on service-oriented architecture (SOA) is proposed. To support network customization, in the middleware layer, a decision algorithm is proposed. It is used for generating the configuration file according to the application requirement. This service provisioning middleware would serve as a generic model for adapting to the required network environment.

**Keywords** Wireless sensor network · Service-oriented architecture · Middleware · Service provisioning

S. Sasirekha (✉)
SSN College of Engineering, Chennai 603110, India
e-mail: sasirekhas@ssn.edu.in

S. Swamynathan
Anna University, Chennai 600025, India
e-mail: swamyns@annauniv.edu



S. Das et al. (eds.), *Proceedings of the 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA) 2015*, Advances in Intelligent Systems and Computing 404, DOI 10.1007/978-81-322-2695-6_28

## 1 Introduction

Wireless sensor network (WSN) in the recent past has witnessed an explosive growth in connecting the digital realm to the physical world [1]. This is mainly due to the technology innovation in microprocessors, wireless communication, and its integration with the microelectromechanical system (MEMS). As WSN consents a fine-grained environment observation at an economical cost much lower than other known monitoring systems such as remote sensing, nowadays it plays a significant role in building large-sized WSNs for most currently evolving practical applications. These sensor networks provide robust service even in toxic and inaccessible regions to humans. WSN has great potential to efficiently plan and coordinate among the nodes to acquire information for real-time scenarios such as handling emergency, military, and disaster relief operations [2]. It brings a new and wide perspective for monitoring physical conditions like temperature, pressure, humidity, etc.

Nowadays, sensor nodes are available at low cost and are miniature in size. Due to their size, sensors have limited energy, storage, communication, and computation capabilities. Hence, they are usually deployed in large arrays to collaboratively extract the environmental data [2]. A WSN is designed to transmit the sensed data from an array of sensor nodes to a data repository on a server or directly to the user through wireless communication. When the sensor network is deployed in a hostile environment, it operates unattended for a long time with sensors equipped with limited battery power. Therefore in these situations, reducing the energy consumption to prolong the network lifetime stands as a critical issue. As most of the energy is spent on transmitting the data than other tasks in WSN, various communication protocols specific to WSN were proposed to handle efficient data transmission over the network. Some of them are LEACH (Low-Energy Adaptive Clustering Hierarchy), PEGASIS (Power-Efficient GAthering in Sensor Information Systems), TEEN (Threshold Efficient sensor Network protocol), MECN (Minimum Energy Communication Network), GAF (Geographic Adaptive Fidelity), and GEAR (Geographic and Energy-Aware Routing). Most of these solutions are dedicated to power efficient routing, focusing on short-range communication and adopt some aggregation mechanism to reduce the amount of data to be transmitted [3].

Some of the related works, as discussed in [4], highlight the importance of involving the application users in the WSN communication process. They also clearly state that it reduces the computation cost of the network with respect to application-specific optimization. Every application class satisfies a specific need and it requires a particular type of communication protocol. Thus, it is understood that the main issue is in providing the most suitable protocol to each application class. However, in general, it is difficult for application developers to choose the protocol that best meets their application needs. Therefore, in this work the main objective is to improve the performance of system by providing application users

with a choice of the communication protocol, from which users can choose according to the application requirements [5, 6]. In spite of the advantages, the existing middleware [7–9] have been built with a high degree of dependency between applications and the underlying communication protocol. Accordingly, the framework also needs to afford a decision unit to provide a choice of suitable communication protocol.

Hence, in this paper a service provisioning middleware is proposed. It acts as a mediator between applications and the WSN. It helps in translating the application requirements to an efficient choice of network configuration and protocols. The middleware is generic enough to be used over a wide range of applications. The key focus of this middleware is interpreting the application needs and selecting the precise network protocol to configuration for performing efficient data aggregation. Besides, it also provides a high level graphical interface for subscribing, publishing, and collecting the sensor data. From the user's point of view, the system offers an abstraction layer to the applications developer from the sensor layer.

This paper is organized as follows: the related work is presented in Sect. 2; in Sect. 3 detailed descriptions of the different layers of the architecture are given; in Sect. 4, the operation of a middleware with a testing environment is illustrated; and finally in Sect. 5, some concluding remarks are given.

## 2 Related Works

In this section, the state-of-art WSN middleware are examined and discussed by classifying them as non-adaptable and self-adaptable middleware in the context of the proposed work. The classification was done based on the support for network reconfiguration at runtime. Here, some of the works related to self-adaptable middleware are discussed. Most of the self-adaptable middleware are based on the service-oriented concepts handling the heterogeneity and composition of the sensor nodes. They manage the adaptation issues based on the structural and behavioral reconfiguration actions. These existing service-oriented architecture (SOA)-based middleware can further be categorized based on the service deployment level. The services deployment basically can be on any of the three distinguished layers such as on the node, on the gateway, or on the base station layer. Middleware such as Tiny Web Services [10] and TinyWS [11] deploy services at the sensor nodes. They interact horizontally between the neighboring nodes and reform the structure according to the specification provided by the adaptive services. Tiny Web Services is an event-based middleware which provides mini Web Services operated by a small version of Transmission Control Protocol/Internet Protocol (TCP/IP) called µIP. TinyWS provides an opportunity to communicate directly with the sensor nodes without going through gateway by embedding Web Services (WS) on the sensor nodes using TCP/IP. Middleware Linking Applications and Networks (MiLAN) is another middleware for WSN, where a description of application

requirement is received from the users and the best sensor protocol and network configuration is chosen. The work presented in [12] belongs to service deployment at the base station layer. It provides an Open Services Gateway initiative (OSGi)-based service-oriented middleware which uses a packet forwarder for communicating between the sensor nodes and base station. However, it requires a system administrator intervention to add a wrapper to communicate with the heterogeneous nodes. One other middleware as discussed in [13] provides a cross-level architecture. It provides a layered approach that deploys services in all the three layers and uses the DPWS for creating WS. However, no adaption is proposed to deal with service failures. At the base station level, the sensor network is also proposed as a database by deploying query proxies inside the network [14, 15]. In [16], a declarative language is proposed, which receives the query submitted by the user and submits it to the sensor networks. COUGAR provides a virtual database concept and a data centric routing approach [17].

All the above discussed middleware are advantageous in the defined context and aim at maximizing the network lifetime. However, the major shortcoming of the above discussed middleware is, they do not provide an abstract representation for the user to specify the application needs and how the sensor data has to be generated. Moreover, these middleware use eXtensible Markup Language (XML) and Simple Object Access Protocol (SOAP) standards for its communication, which stimulate an overhead and consume a large part of the device resources [18]. The main difference between the existing middleware and the proposed work is, here, a customary depiction of the application requirements and sensor data generated are provided. Also, the proposed system relies on XML and REpresentational State Transfer (REST) standards retaining the device resource for a much longer time. Hence, the proposed system aims at providing an adaptable WSN configuration and protocol support for every specific application requirement.

## 3 System Model for Data Provisioning Middleware

The proposed middleware serves as a uniform, standard, and abstract development model for developing WSN applications. This framework is developed by reflecting the concepts of service orientation specific to WSN [19]. The architecture includes three significant layers such as the sensor network layer, data processing layer, and the user interface layer as shown in Fig. 1. The sensor network layer is built by deploying several sensor nodes which have different sensing capabilities. Within the same network one powerful node with greater processing and storage capacity is deployed to act as sink node. The sensor network layer transmits the sensor data and the network management information to the outside entities via the sink node using the corresponding providers. These sensor data and other information received from the network layer are complied, classified, and stored for future use. Then the middleware-specific sink node in the network layer is used to pass request to the

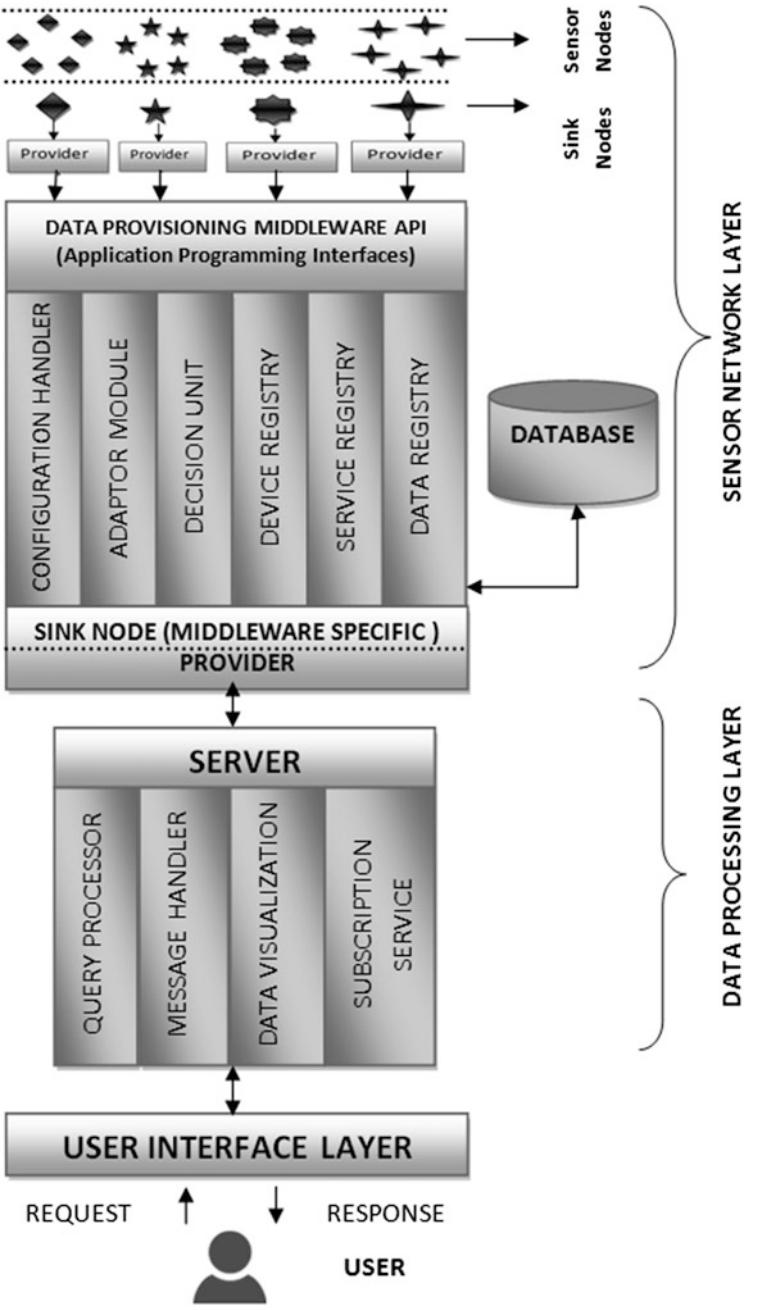


**Fig. 1** Design of data provisioning middleware

network. The entire request to the data and network control are made using the WS [15] running in the server, which is present in the data processing layer. These WS process the data received as response for the request sent from the sensor network layer. Then the user interface layer is provided with various options where the request can be made and data can be monitored and visualized for future analysis. Since the SOA concepts are reflected in this architecture design, certain assumptions are made. Each individual sensor node is considered as the WS provider; the sink node is considered to act both as service provider for the users' request and as a service requester to get the data from the network layer. In order to handle the heterogeneity of the sensor nodes and to improve the efficiency of the overall system a middleware layer is interfaced in between the network and the data layer. It also provides an adaptable configuration service to manage the network behavior at execution time more efficiently.

The middleware deployed in WSN section provides an interaction layer between the application and sensor networking layer. It provides a high-level depiction for the application users, the sensor field, and to perform efficient WSN data aggregation. Sensor nodes in the WSN that act as service providers register and publish themselves with the network information in the Service Registry. A default multi-hop (data centric) protocol defined in the network forwards all these messages toward sink nodes that push to the registry. A control message is exchanged among them to keep their database updated and the latter can be used to coordinate among the sensors in the network. Here, the system adopts XML to describe the interface and support and language encoding. For the protocol support, REST [20] is adopted and is used for formatting XML messages and transporting invocations between sensors by using the underlying default network protocol. Then the services provided by the middleware are achieved by deploying application programming interfaces (APIs) for sink node. The middleware design portrays the implementation scenario where, the network layer comprises a heterogeneous sensor environment with various sensing capabilities. The middleware components are deployed in the sink nodes. Then a registry and server are maintained discretely for user access.

## *3.1 Design of Sink Node Middleware APIs*

Users of the WSN are often proficient in their application domain but not much in networks, hence the configuration and WSN operations are preset [21, 22]. In such a scenario, if the user wants choose the network protocol and reorganize according to the application requirement it would be difficult. Therefore, to guide the users with the best choice of network configuration for a specific application, an automatic decision process is offered. This choice is based on results of previous works carried out in the field, reported by researchers. These works are studied and stored in the database. The adaptation module of the middleware holds this database with all the historical information about the different WSN configuration choices. In

addition, the average values for the various performance metrics, such as delay and accuracy, for each executed application are included in the database for the corresponding network configuration. This information is used as the major input by the decision algorithm of decision unit to provide the best choice of network configuration. However, based on factors such as the number of sensors in the data source, number of sink nodes, and the choice of data delivery model, the decision algorithm decides the dissemination and network topologies as shown in Fig. 2.

For every request made by the application providers the decision unit API runs the decision algorithm. The decision algorithm generates the corresponding configuration file. The generated file is pushed to the configuration manager API. Receiving the configuration file, it processes them and using the adaptor module API, the configuration API controls the hardware and topology of each individual sensor network through its corresponding sink nodes. These API modules communicate among them with the XML messages to receive and process the application input messages. Also, the middleware has an XML-based API for sending

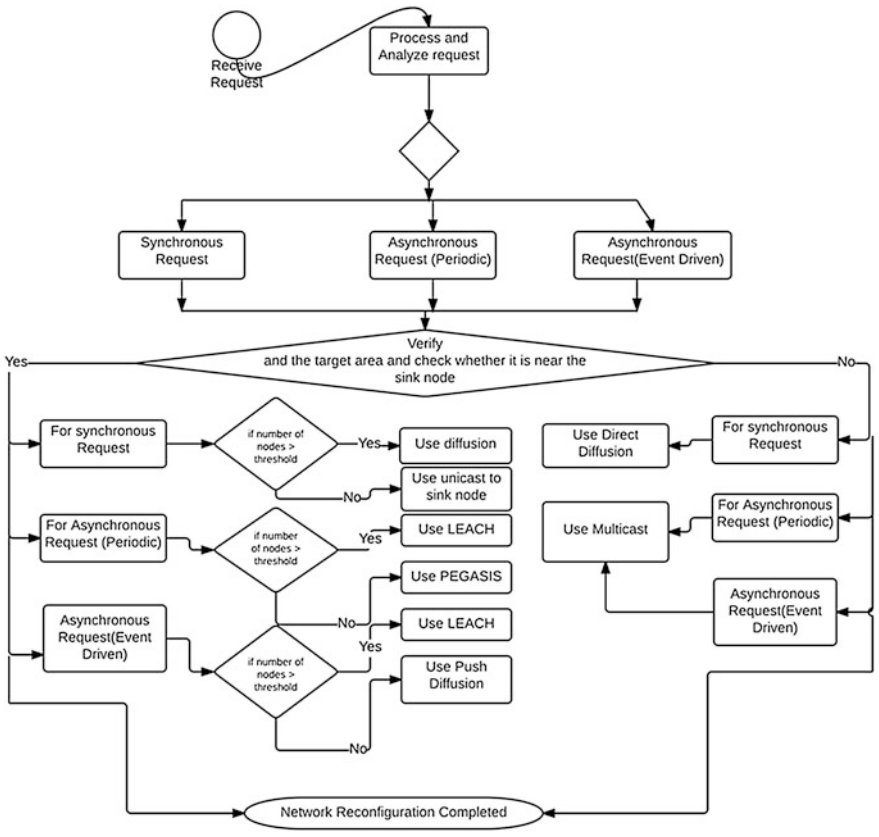


**Fig. 2** Flow chart of decision algorithm

the configuration file to configuration handler of node level API and it activates the dissemination protocol according to the current WSN application. The decision algorithm decides the strategies of data dissemination and network topologies based on the request type submitted by the user, the size of the target area, and the expected number of data sources and sinks are considered. From the choice based on the previous works, the possible combination of network topology and dissemination strategy was identified. It was inferred that, in general, the network topology to support energy efficient data dissemination can be classified as flat and hierarchical. Further, according to the application requirement they are merged with dissemination algorithm such as direct diffusion, push diffusion, one-phase diffusion, unicast, multicast, and clustering with LEACH or PEGASIS.

Then the infrastructural information of the WSN and users are held in the registry. The various services related to the network are maintained in the form of various sub-registries. The aggregated network information from sink node service provisioning API is registered in the device registry. Service information such as the names of the methods, the order, number, and types of their parameters, and the return types along with meta-data information are maintained in the service registry The service registry provides an option for the user to subscribe a data delivery model required by the application: synchronous, asynchronous (periodic-running, event-driven), which is further processed by the sink node APIs to generate the configuration file to establish the WSN operation. The sensor historical data API is responsible for maintaining the past records of the sensors, inclusive of previously sensed data and status of the node through the network lifetime.

## *3.2 Design of Server Layer*

This layer acts as WS provider, abstracting the networking layer and making available a simple WS. It provides an interface to view the various services offered by each of the networks and also a provision to check and consult the registry to register for events and maintenance tasks. Once the application is aware of the available WSN services provided by the network, the user can subscribe to the services through subscribe messages. The parameters of the request message vary depending on the request type. For instance, in case of synchronous message, the sensor type and the geographical coordination of the target area are provided. In case of a periodic-running query type, the sensor type, the target area, the duration of data acquisition rate, and duration are mentioned. Finally, incase of event-driven queries the sensor type, the target area, and the event to be monitored have to be specified. Applications may also request the use of aggregation function which will also be passed in the configuration file. The query processor services process the application requirements and message handler translates the request and coordinates with the service registry and the sink node in generating the configuration file to set up the underlying communication layer.

### *3.3 Configuration File Generation*

As shown in Fig. 2, each application provider submits the request query to the decision algorithm along with various other parameters such as the size of the target area, the estimated number of data sources, and sinks. These parameters have a significant role in deciding the different strategies of data dissemination and network topologies. Initially the algorithm verifies for the request type submitted by the provider. The request query is analyzed and determined if the query type is a synchronous or asynchronous periodic or event-driven query. The network reorganization is mainly based on this request type. Next, for all the request types, the size of the target area and the number of nodes near the sink node is verified. It is analyzed for, if the number is greater than a defined threshold. In case it is large, for a synchronous request again, the total number of nodes in the target is verified against a defined threshold, if it is has a large number of nodes, diffusion dissemination strategy is chosen, otherwise unicast to sink node is chosen. Similarly, it is also verified for asynchronous periodic and event-driven query; in this case if the number of nodes is large, LEACH dissemination strategy is chosen, otherwise PEGASIS and push diffusion is chosen respectively. In case it is less, for a synchronous request the direct diffusion dissemination strategy is chosen. Similarly, for asynchronous periodic and event-driven query muticast dissemination strategy is chosen.

Further, the middleware passes the decision chosen to the network protocol, which in turn triggers the needed infrastructure for communication. Finally the message is propagated to sensor nodes, based on the chosen strategy for data dissemination. If the request sent from the application matches its data type it responds with the data delivery message.

Therefore the proposed middleware operates in a service-oriented way providing a flexible, adoptable, generic system for programming abstractions and positioning them right from the hardware platforms up to end-user applications in a heterogeneous WSN.

## 4 Testing Environment

A WSN environment is created with 4 Arduino Mega Board interfaced with Digi XBee using XBee shield to form a mesh network as shown in Fig. 3, to monitor the temperature and humidity of the testing environment. To build sensor network with a mixture of hardware platforms, Digi XBee is used to communicate because they are 802.15.4 compliant. Since tuning of many parameters of the XBee is very versatile, XBee is preferable to CC2420. Also, XBee module has an IEEE 64-bit address and has a provision to configure in the AT mode or API; as the aim is to

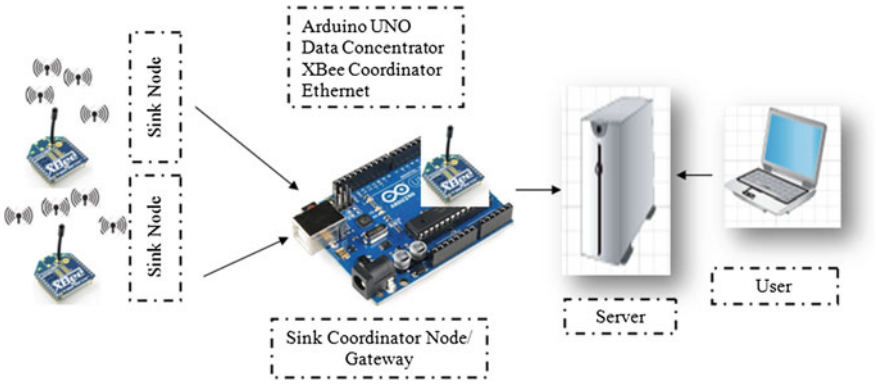


**Fig. 3** Testing environment

configure the network from an API, we choose an API mode. The XBee modules come preloaded with the ZNet 2.5 firmware which implements the ZigBee protocol stack. XBee has its own microcontroller inside and for programming it uses the software called XCTU. This microcontroller is used to program the routing and other communication details of the XBee.

In order to make all the sensor platforms to communicate, some basic settings have to done in the network configuration level. First of all, for the radios to communicate, it is mandatory to set the same Personal Area Network (PAN) ID and sync them in the same channel. For the purpose of testing it is assumed that PAN ID 0x1234 and channel 0x0D are set for all the radios. Once the communication aspects are programmed among the various sensor platforms the sensor nodes are deployed. Then the middleware APIs are installed in the sensor and sink nodes to provide the abstraction. The necessary implementation codes for the sensors to implement them as services are defined. This implementation is platform-dependent, that is, the application programmer uses C++ to write Arduino programs. The users can access the WSN through the WS deployed in the server. The central server acts as a gateway to the WSN. The intelligence level of the WSN increases upstream in the network. Some of the data collected are shown in Fig. 4.

To address the improvements of the above stated middleware system, it is demonstrated that when using a network routing protocol chosen according to the application needs the network utilization is proven to be high. A comparison is run with middleware with fixed data centric protocol and an adaptive middleware where the network routing protocol changes according to the application need. It is initially fixed with multi-hopping flat network routing protocol, Sensor Protocols for Information via Negotiation (SPIN). The idea behind SPIN is to name the data using high-level descriptors or meta-data. Before transmission, meta-data are exchanged among sensors via a data advertisement mechanism. Each node upon receiving new data advertises it to its neighbors and interested neighbors, i.e., those that do not have the data, retrieve the data by sending a request message. The

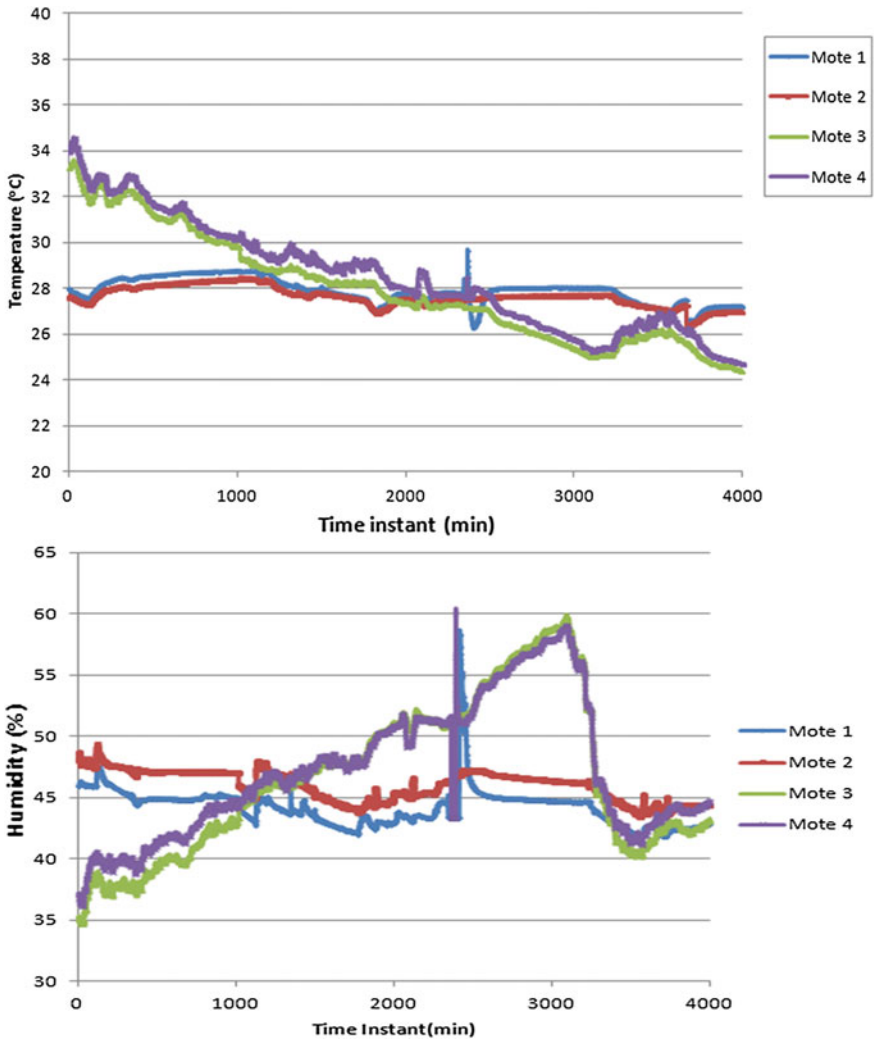


**Fig. 4** Sample temperature and humidity data collected from 4 motes

adaptive network routing protocols for the data delivery model are chosen based on the decision algorithm explained in Configuration file generation section.

The results are generated for both the scenarios, the fixed routing protocol and adapted routing protocol for the three data delivery models—Synchronous, Asynchronous Event-Based, and Periodic Data Set models. The adaptive network routing protocols for the data delivery model are chosen based on the decision algorithm explained in Configuration file generation section.

**Table 1** Performance measures of the proposed system

| Data delivery model | Routing protocol | No of Hop counts (nos.) | Total energy dissipation in nodes (J/Msg.) | Latency (ms) |
|---|---|---|---|---|
| Synchronous data set collection | SPIN | 5 | 5.5 | 1000 |
| | diffusion | 2 | 8.5 | 400 |
| Asynchronous event data set collection | SPIN | 5 | 5.5 | 1000 |
| | LEACH | 1 | 9 | 200 |
| Asynchronous periodic data set collection | SPIN | 4 | 5.5 | 800 |
| | Pull diffusion | 1 | 9 | 200 |

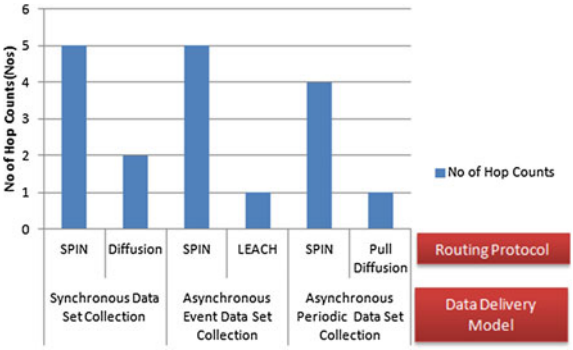


**Fig. 5** Number of Hop counts

In the network setup, 4 sensor nodes (also called as motes) are deployed. The initial energy level in all the nodes is set as 10 J. After deployment, the user request is initiated and the query processor processes according to the application requirements. The message handler translates the request and coordinates with the service registry and the sink node is involved in finding the data delivery model and generating a configuration file to set up the underlying communication layer.

The process is repeated for different data delivery models and the performances of this model are monitored. The number of hops it takes to reach the sink node, total energy dissipated in the nodes while transmitting, and based on the hop count the latency is measured. The generated results are tabulated in Table 1 and graphically plotted as shown in Figs. 5, 6 and 7.

For simplicity of energy analysis, a first-order radio model [23, 24] is adopted. Energy consumption in circuitry for running the transmitter or receiver and in radio amplifier for wireless communication are $E_{circuitry} = 0.5$ J/msg and $E_{amplifier} = 0.5$ pJ/msg, respectively. The value of $E_{amplifier}$ is directly proportional to the square of transmission distance. Therefore, using the following formulas (1) and (2):

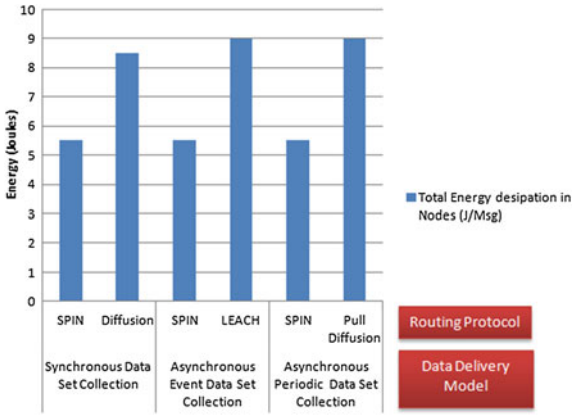


**Fig. 6** Energy dissipation in the nodes

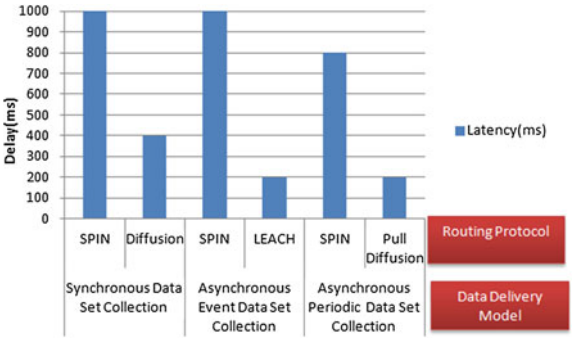


**Fig. 7** Latency in transmission

The energy consumed for transmitting a single packet is

$$E_{\text{transmit}}(k,d) = E_{\text{circuitry}} \times k + E_{\text{amplifier}} \times k \times d^2 \quad (1)$$

The energy consumed for receiving single a packet is

$$E_{\text{recieve}}(d) = E_{\text{circuitry}} \times k \quad (2)$$

where k is the size of transmitted packets, and d is the distance between a transmitter and a receiver.

There is difference in the performance of the various data delivery models. The performance requirements of the model are application-specific. The user should be able to adapt or choose the required network routing protocol. The proposed middleware architecture supports the selection of required routing protocol for the data delivery model by the user. Hence the model is flexible to the user's specification. The analysis on the performance measures aids to infer that the overall system performance seems to be improved.

## 5 Conclusion

In this paper, a service-oriented approach-based service provisioning middleware for WSN was presented. It provides a flexible and generic platform for application developers to perform operation on WSNs. This middleware layer abstracts the WSN infrastructure and protocol information from the developer, as most of the existing middleware have been built with a fixed network topology and a data dissemination strategy to perform operation of WSN; whereas the network design scenario varies depending on the application context. Hence in this paper, the proposed service provisioning middleware provides an extended support for customizing the network configuration for an application requirement with respect to its context. For this purpose, a decision algorithm was proposed. It decides on the configuration required for application context based on the type of request query, the size of the target area, and the estimated number of data sources and sinks. Thus, adopting this system it provides an efficient usage of the WSN thereby extending its lifetime. The middleware was tested to handle three basic request types such as synchronous, asynchronous event-based, and asynchronous periodic-based requests. It is inferred from the results that it chooses the best suitable option, based on information provided by the users' interest. This proposed work takes an initiative to build energy efficient WSNs.

## References

1. Khemapech, I., Duncan, I., Miller, A.: A survey of wireless sensor networks technology. In: 6th Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting, pp. 1–42. Liverpool, UK (2005)
2. Rawat, P., Singh, K.D., Chaouchi, H., Bonnin, J.M.: Wireless sensor networks: a survey on recent developments and potential synergies. J. Supercomputing **68/1**, 1–48 (2014)
3. Akkaya, K., Younis, M.: A survey on routing protocols for wireless sensor networks. Elsevier Ad Hoc Netw. J. **3**(3), 325–349 (2005)
4. Heidemann, J., Silva, F., Estrin, D.: Matching data dissemination algorithms to application requirements. In: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, pp. 218–229, USA (2003)
5. Heinzelman, W., Kulik, J., Balakrishnan, H.: Adaptive protocols for information dissemination in wireless sensor networks. In: Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99), pp. 1–15, WA (1999)
6. Wang, M., Cao, J., Li, J., Dasi, S.K.: Middleware for wireless sensor networks: a survey. J. Comput. Sci. Technol. **23**(3), 305–326 (2008)
7. Heinzelman, W., Murphy, A.L., Carvalho, H.S., Perillo, M.A.: Middleware to support sensor network applications. IEEE Netw. Mag. Special Issue **18**(1), 6–14 (2004)
8. Mohamed, N., Al-Jaroodi, J.: A survey on service-oriented middleware for wireless sensor networks. J. IEEE Int. Conf. Serv. Oriented Comput. Appl. **5/2**, 71–85 (2011)
9. Hadim, G. Mohamed, N.: Middleware challenges and approaches for wireless sensor networks. IEEE Distrib. Syst. Online **7/3**, 1–23 (2006)

10. Nissanka, B., Priyantha, A.K., Goraczko, M., Zhao F.: Tiny web services: design and implementation of interoperable and evolvable sensor networks. In: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, pp. 253–266. ACM, USA (2008)
11. Othman, N.Y. Glitho, R.H., Khendek, F.: The design and implementation of a web service framework for individual nodes in sinkless wireless sensor networks. In: 12th IEEE Symposium on Computers and Communications, pp. 941–947 (2007)
12. Prinsloo J.M., Schulz C.L., Kourie, D.G., Theunissen, W.H.M.: Strauss, T., Van Den Heever, R. Grobbelaar, S.: A service-oriented architecture for wireless sensor and actor network applications. In: Proceedings of the 2006 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries (SAICSIT '06), pp. 145–154 (2006)
13. Leguay, J., Lopez-Ramos, M., Jean-Marie, K., Conan, V.: An efficient service oriented architecture for heterogeneous and dynamic wireless sensor networks. In: 33rd IEEE Conference on Local Computer Networks, pp. 740–747 (2008)
14. Devices Profile for Web Services (DPWS) specification. (2006). http://schemas.xmlsoap.org/ws/2006/02/devprof/
15. Bonnet, P., Gehrke, J.E., Seshadri, P.: Towards sensor database systems. In: Proceedings of the 2nd International Conference on Mobile Data Management, pp. 3–21, Hong Kong (2001)
16. Govindan, R., Hellerstein, J.M., Hong, W., Madden, S., Franklin, M., Shenker, S.: The sensor network as a database. USC Computer Science Department Technical Report, pp. 02–771, (2002)
17. Yao, Y., Gehrke, J.E.: The cougar approach to In-Network query processing in sensor network. Sigmod Record. **31**(3), 9–18 (2002)
18. Delicato, F., Pirmez, L., Pires, P., de Rezende, J.: Exploiting Web technologies to build automatic wireless sensor network. In: 8th IFIP IEEE International Conference on Mobile and Wireless Communication, vol. 211, pp. 99–114 (2006)
19. Delicato, F.C., Pires, P.F., Pirmez, L., Carmo, L.F.: A service approach for architecting application independent wireless sensor networks. Cluster Comput. **8/2–3**, 211–221 (2005)
20. Graham, S., et al.: Building Web services with Java: making sense of XML, SOAP, WSDL, and UDDI. Sams Publishing (2002)
21. Fok, C.-L., et al.: Adaptive service provisioning for enhanced energy efficiency and flexibility in wireless sensor networks. Sci. Comput. Program. **78/2**, 195–217 (2013)
22. Chandrakant, N., Tejas, J., Harsha, D., Deepa Shenoy, P., Venugopal, K.R., Patnaik, L.M., Chancellor, V: EMID: maximizing lifetime of wireless sensor network by using energy efficient middleware service. In: International Conference on Intelligent Information Networks, pp. 314–317 (2011)
23. Lindsay, S., Raghavendra, C.S., Sivalingam, K.M.: Data gathering in sensor networks using the energy delay metric. In: Proceedings of the 15th International Parallel & Distributed Processing Symposium, p. 188 (2001)
24. Tang, F., You, I., Gou, S., Gou, M., Ma, Y.: A chain-cluster based routing algorithm for wireless sensor networks. J. Intell. Manuf. **23**(4), 1305–1313 (2010)