Telecommun Syst (2013) 53:61–68
DOI 10.1007/s11235-013-9677-2

A service-integrated sensor network middleware applied to industrial solutions of IoT related

Yingyou Wen · Zhi Li

Published online: 22 June 2013

Abstract With the deployment and application of Internet of Things, middleware for sensor network has become an increasingly important factor to facilitate the programmer task and bridge the gap between the applications and the underlying infrastructure. Most existing middleware for sensor networks lack attention for integrating services into a generic architecture and present inflexibility that make them useless in the context of industrial solutions of Internet of Things. To solve this problem, we proposed a new middleware deployment model and present layered and distributed architecture utilizing our "Ends to the Middle" approach. Actual application of middleware demonstrates that our proposed architecture is highly modular and efficient, offers good performance in complex application scenarios of Internet of Things.

Keywords Sensor networks · Middleware · Architecture · Gateway

1 Introduction

After more than ten years of development, sensor network has been received extensive attention, and become one of the most competitive fields of applied technology [1], also an important technology basis and means of IoT (Internet of Things) related applications. However, deployment of sensor network and application development is still a big challenge owing to those well known factors such as changing network topology, diverse communication protocols, heterogeneous sensors and complex coding of node level. Application developers not only need to pay attention to requirements of user, but also master many underlying technology such as data acquisition, time synchronization, redundancy control, topology management and security, which greatly increased difficulties of application development, especially those large-scale industrial application solutions of IoT related [2].

Sensor network middleware is a software platform located between the application and the underlying sensor network [3]. Middleware provide developer with a unified view and development interface, shielding the underlying complex structure of the sensor network, making application development of sensor network become more simple and efficient. With the rise of the concept of IoT, sensor network middleware has become an important aspect of practical sensor network application, which restricts the process and scale of sensor networks applied to industry solutions of IoT related.

Although many researches have been done on sensor network middleware, it seems that most of existing sensor network middleware architecture is designed according to the function of the sensor network itself, but not from the perspective of application development needs. These middleware can be used in a specific application scenario or a small application, but not those large-scale applications applied to industry solutions of IoT related.

To solve this problem, we present new middleware architecture during our implementation process of large-scale

The authors gratefully acknowledge the support of Neusoft Research.

Y. Wen (✉)
Northeastern University Research, Northeastern University, Shenyang 110000, China
e-mail: wenyy@neusoft.com

Z. Li
College of Information Science and Engineering, Northeastern University, Shenyang 110000, China
e-mail: zhi-li@neusoft.com

industry solutions of IoT related, which provides a centric interface abstraction to fulfill the requirements of application development.

The rest of the paper is organized as follows. In Sect. 2, we present some related works and analyze their merit and deficiency. Section 3 presents a design of our sensor network architecture. Section 4 gives the implementation details and a brief summary are discussed in Sect. 5.

2 Related works

Sensor network middleware is a development platform and infrastructure designed for the characteristics of sensor network. Its main role is to isolate the physical network from upper layer applications, shield differences from various manufacturers in communication protocols and data formats, provide a unified data processing, network monitoring and service dispatch interface for the application development.

As existing middleware technologies have different characteristics, many literatures give multiple classifications, such as database-inspired, virtual machine, mobile agent, event driven and so on [4]. These middleware classifications specify the differences in middleware design, which reflects different understandings of the researchers about roles and tasks of sensor network middleware. Among these classifications, there is a thought through the deployment position of middleware to distinguish; the middleware can be divided into node-level, gateway-level and client-level [5]. By using this kind of approach, we can see that most of current implementations of middleware belong to node-level middleware such as Mate [6], TinyDB [7] and Agilla [8] etc., which try to solve practical problems of application development in the node-level by focusing on data collection, transmission, energy saving and re-programming ability of nodes. However, the requirements of application developers have not been given full consideration in node-level middleware, so it is not conducive to the actual use, especially to those applications in large-scale industrial solutions of IoT related, because: (i) Most node-level middleware rely on high capability of sensor node, such as requiring the node has a strong computing, storage and communication capabilities, which greatly enhance the manufacturing cost of sensor node and hinder widespread deployment of sensor networks and applications; (ii) Although partly shield the underlying implementation of the sensor node,node-level middleware still have a API closely related with the node operating system and specific coding rules, which can not completely isolated application developers from sensor network, nor support the application developers using a familiar development tools and means to carry out large-scale sensor network application development. (iii) Node-level middleware usually distribute data processing and event monitoring to each node or a node set, but does not provide a unified services and develop interface general enough to upper layer applications. Sensor network can not be developed as a whole by using node-level middleware. In a large industry-oriented applications of IoT related, it is hard to imagine a development of distributed application with complex business logic by programming for each individual node.

Relative to node-level types, middleware on the gateway level tend to expose network functions and services as a whole to the upper application. Garnet [9], GSN [10], Milan [11] are all belong to this type. Gateway-level middleware has the advantage of providing more abstract and uniform interface for the upper layer application development, but lack of support for fine-grained application development and node-level programming. Moreover, most of the current projects on gateway-level middleware are at an early stage and present important nonnegligible drawbacks such as weakness in architecture scalability, simple function and lack of effective integration of nodes and gateway, which make them useless in the context of new emerging applications of IoT related.

As can be seen from the above analysis, many existing middleware application can not meet the development needs of IoT related applications. To solve this problem, we proposed a generic middleware architecture which aims to help application developers to build industrial applications of IoT related exactly matches user requirements, this architecture take into account of advantages of both node-level and gateway-level.

3 Architecture designment

Architecture has an important effect on middleware functional model, which determines flexibility, scalability and enforceability of middleware.

To carry out the design of middleware architecture, we must first have a clear role description and definition of the deployment model. As can be seen as following, our proposed middleware architecture is different from node-level and gateway-level middleware, it is a combination of the two.

3.1 Deployment model of middleware

The term middleware itself means different things depending on the computing field it's used for. In distributed systems, middleware is usually defined as software that lies between the operating system and applications running on each node of the system.

In fact, most of existing node-level sensor middleware is designed according to this definition and located between node operating system and applications. But, just as discussed above, this kind of middleware is not as useful as it have been imagined to be.

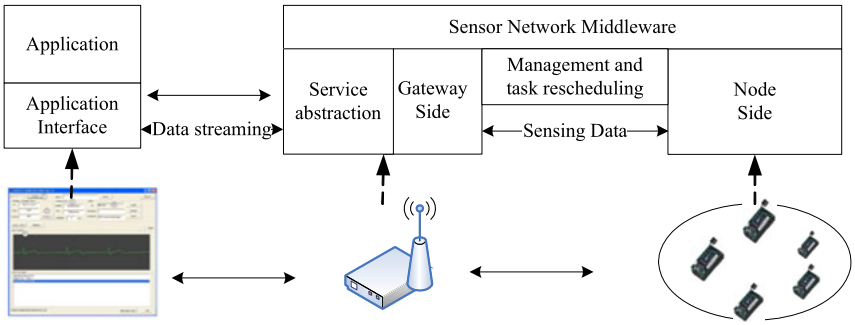

Fig. 1 Deployment model of middleware

To fulfill the requirements of actual development, we believe that sensor network middleware applied to large-scale networking and industrial applications of IoT related should be located between "sensor network and business applications" instead of "sensor node operating system and applications". Only in this way, details of the underlying network and constraints of devices related can be effectively shielded from application developers, which enable application developers to concentrate fully on realization of business logic and handling of data and event.

Nevertheless, implementation of partial middleware functionality on top of the node operating system is also necessary in order to support fine-grained application development and deployment requirements. In other words, sensor network middleware should be able to provide not only the services and unified application interface, but also means to support reprogramming and task reallocation of the underlying network node.

Taking all these considerations above, a deployment model of our proposed middleware is shown as Fig. 1.

The middleware consists of two parts, one is deployed on node side and the other deployed on gateway side. As an intermediary between application system and underlying sensor network, gateway side providing services abstract and development interface for the application. Through control and scheduling of gateway, node side implements the task reallocation to support the application of fine-grained implementation.

This deployment model is more practical to be used in development of large-scale networking and industrial applications for the Internet of Things, with flexibility to facilitate the programmer task in both scenarios of heterogeneous and homogeneous.

3.2 Principles of architecture designment

Existing design principles of sensor middleware architecture can be divided into two basic types, "Bottom-up" and "Top-down". In "Top-down" mode, middleware architecture is designed from the application point of view. Functional requirements of s sensor network middleware platform to provide services is to determined according to the application needs, which is realized by the underlying sensor network nodes' functions and services. Cougar, SINA, COSMOS can be classified into this mode. This kind of middleware for sensor networks is mostly designed for specific applications, so the middleware should fully consider the needs of specific applications.

Different from "Top-down" mode, "Bottom-up" middleware is designed fully considering characteristics of sensor node. Functions and modules of middleware are determined by analysis of node configuration, mobility, data types, communication protocols, node distribution, etc. In general, middleware based on this approach are required to provide publish and subscribe mechanisms to make services available to specific applications. "Bottom-up" mode does not target specific applications, and only concerned with their own sensor network capabilities to offer. In some cases, there are limits to this approach in fulfilling development needs for the absence of specific applications considerations

Referring to different design methods and some specific middleware architecture design, considering the goals and deployment model; we adopt an "Ends to the middle" method to configure our middleware architecture for sensor network, just as showed in Fig. 2.

First, based on a variety of application development requirements, we give full consideration to the type of service provided by the middleware and application development support capabilities, then focus on solving the classification of services, definitions, descriptions. Secondly, considering characteristics and the base capacity of underlying sensor network, we achieve functional classification and basic module division and descriptions. At last, through combination of upper services abstraction and functional definition, mapping between the two ultimately form our whole middleware architecture.

3.3 Architecture designment

In accordance with the aforementioned design principles of "Ends to the middle", sensor network middleware architecture design conduct primarily in three areas: abstraction of application development requirements, capability classification of underlying network and the definition of services and

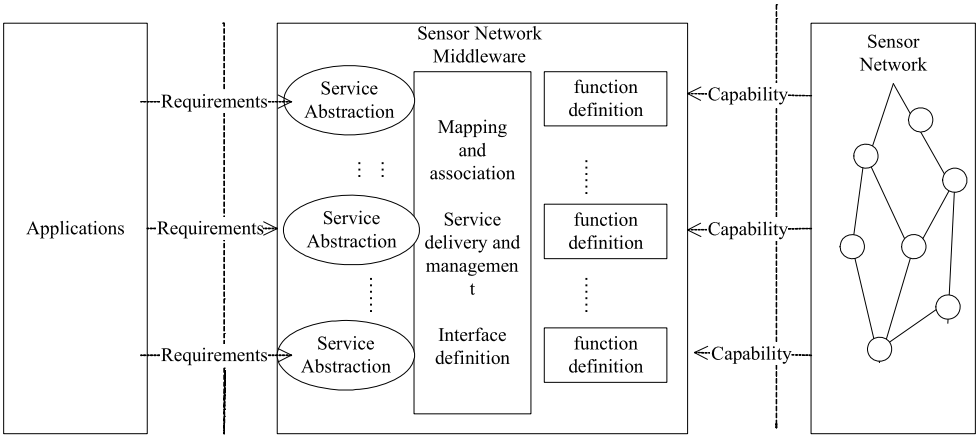

Fig. 2 "Ends to the middle" middleware design process

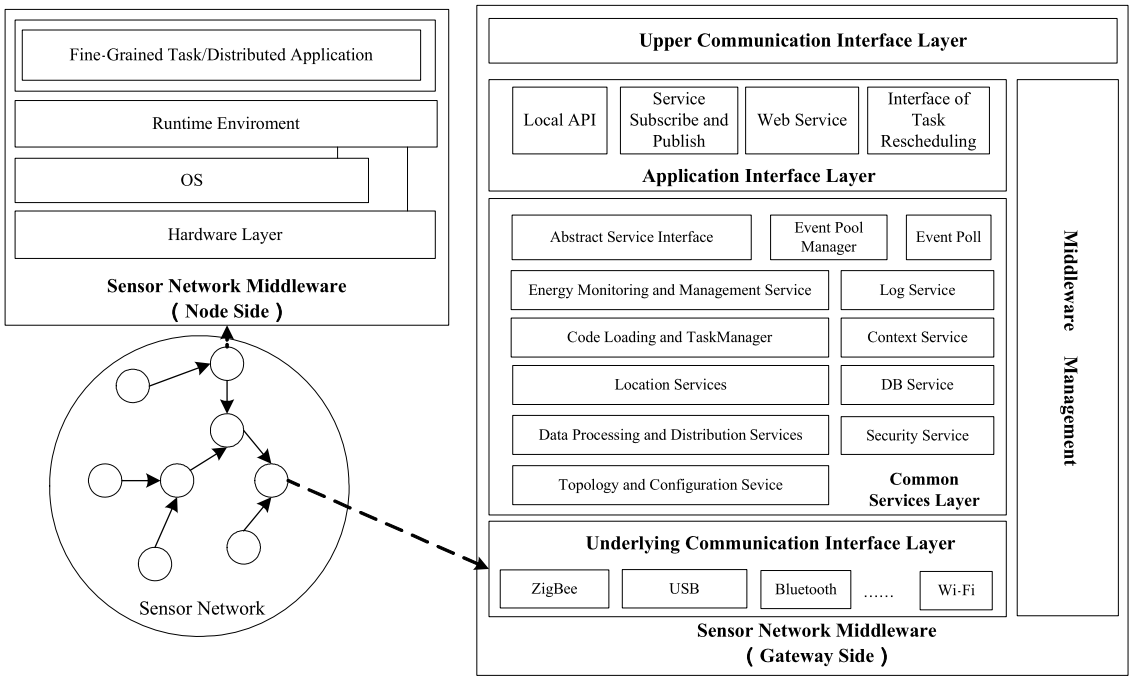

Fig. 3 Sensor network middleware architecture

functions. To meet the requirements of large-scale networking and application development of industry solutions based on Internet of Things, the whole architecture should not only take full account of the needs of business application development, but also take into account the characteristics and the basic capabilities of the sensor network itself, including deployment model and roles of the middleware. Figure 3 below presents an overview of our proposed middleware, and a description of each part is provided

The whole middleware architecture consists of two sides and four levels. To distinguish by deploying position, middleware is divided into two sides, gateway and node, which are independent in location but closely linked essentially.

Gateway side shields details of the underlying network and data processing, also provides a service abstraction interface and application development interface to the upper layer. Node side inherits design objectives of existing node-level middleware, coordinate with the gateway side to realize deployment of distributed application and fine-grained functions on sensor devices. The relationship between node side and gateway side is maintained by means of node reprogramming and task rescheduling, which is provided by a service in our proposed middleware architecture.

From the vertical function partition perspective, the middleware as a whole is divided into four levels in gateway side: the Underlying Communication Interface Layer, the

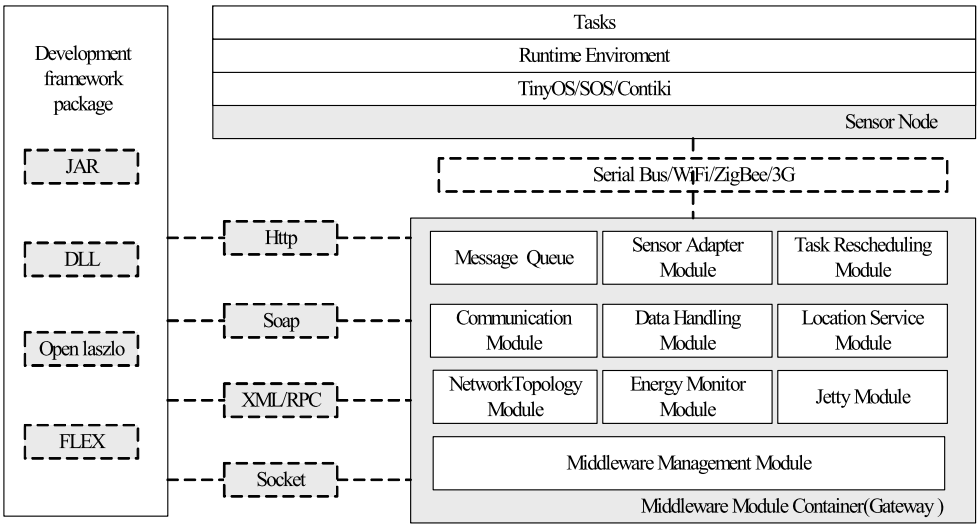

Fig. 4 Implementation of middleware gateway side

Common Service Layer, the Application Interface Layer and the Upper Communication Interface Layer.

The bottom layer of our middleware is called the Underlying Communication Interface Layer. It is mainly responsible for underlying network communication protocol identification and message handling, and achieving reliable communication between middleware and sensor networks.

The intermediate layer is called the Common Services Layer, which realizes mapping and associations of sensor ability and upper service, also implements coordination and handling of calls among various services.

The Application Interface Layer is responsible for the definition of service interface and application development interfaces, and support service calls from upper layer and application development.

The Upper Communication Interface Layer provides information transfer and data delivery to high-level application through wireless communication networks (including 3G/4G/WIFI) and the IP network.

In this architecture given above, the Common Services Layer is the most complex and critical, in which the specific function module and the relationship between them determines the functional model of the middleware.

Role as a connecting link, the most basic and key feature of sensor network middleware is confined to the following three categories: sensor data processing and analysis, sensor network management functions and communication functions. As shown in Fig. 3, we present the basic sensor network middleware services and functional modules. It should be stressed that the goal of our proposed middleware architecture is not to design a large and comprehensive framework to fulfill all application needs, but to consider the actual requirements of sensor network applications development and clarify relationships between different services under various scenarios, so that we can flexibly combinative and choice services needed from a unified point of view, based on the requirements of each specific application, and simplify development and deployment of applications based on sensor network in a complex networking environment.

4 Implementation details

4.1 Gateway side implementation

There are two aspects need to be carefully considered during the implementation of sensor network middleware. On the one hand, component-based system design and implementation will help the sensor network middleware to adapt to different application scenarios and business needs, which will make middleware conducive to be the most widely used in large-scale applications applied to industry solutions of IoT related. At the same time, component-based middleware implementation will provide a flexible basic framework to support the integration of refined and improved features in future. For this consideration, we adopt a component-based framework to realize the proposed generic architecture. On the other hand, taking into account the actual deployment environment of sensor network, gateway is usually limited in the computing and power capacity, also constrained by the equipment specifications. Therefore, gateway side of sensor network middleware should have a flexible embedded software framework, which is suitable for running on small embedded device with resource consumption.

Based on the two considerations above, we use a small embedded system to realize our proposed middleware platform as shown in Fig. 4, which integrates functional components about data processing and network monitoring, including the underlying network communication interface, and the upper layer APIs.

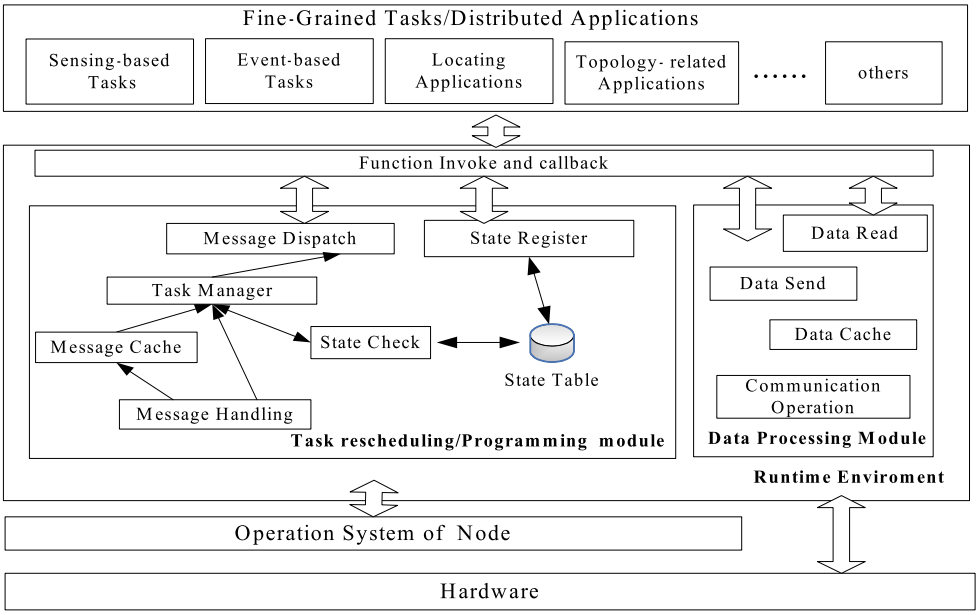

Fig. 5 Structure of node side middleware

4.2 Node side implementation

The node side middleware lies between node's operation system and applications. Our implementation of node side middleware consists of four levels, namely hardware, OS, runtime environment and task/application layer. The structure of node side middleware is shown in Fig. 5.

The main function of the runtime environment is to provide ways and means of node-level programming, shielding the underlying operating system and hardware, simplifying application deployment on node side. This layer has two main modules, Task rescheduling/Programming module and Data Processing module. To better understand all components of node side of our middleware, their functions are explained below.

Task rescheduling/Programming module is mainly responsible for providing programming APIs of node-level and rescheduling of tasks according to instructions from gateway side. There are a lot of functional components created in this module, including Message Receive (MR), Message Cache (MC), Task Manager (TM), State Check (SC) and Message Dispatch (MD). These components cooperate to accomplish task scheduling and support different applications and tasks. MR component receives messages from the gateway. These messages are divided into node-level messages and task-level messages according to the components to be sent in next step. For node-level messages, AM component usually deals with them to reschedule tasks. Received messages are also divided into high priority messages and general messages according to response time limitations of processing. The high priority messages need to be handled immediately. MC component stores messages received since MCU in node hardware can not handle several messages simultaneously. TM component is in charge of messages dispatching and handling of node-level messages, while task-level messages are forwarded to upper layer. When receiving higher priority messages, it handles and forwards them by interrupt mode. By checking the State Table, it operates corresponding tasks according to the message received. State Table keeps state information of tasks and applications, such as start, stop and sampling frequency.

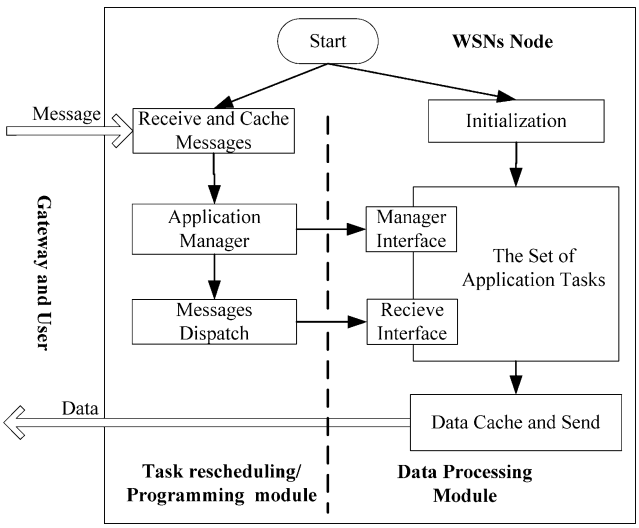

Fig. 6 Workflow of node side middleware

Data processing module consists of data read, data cache and data send. This module provides many APIs for upper layer to invoke, and execute operations of transferring and forwarding of sensing data and other application information.

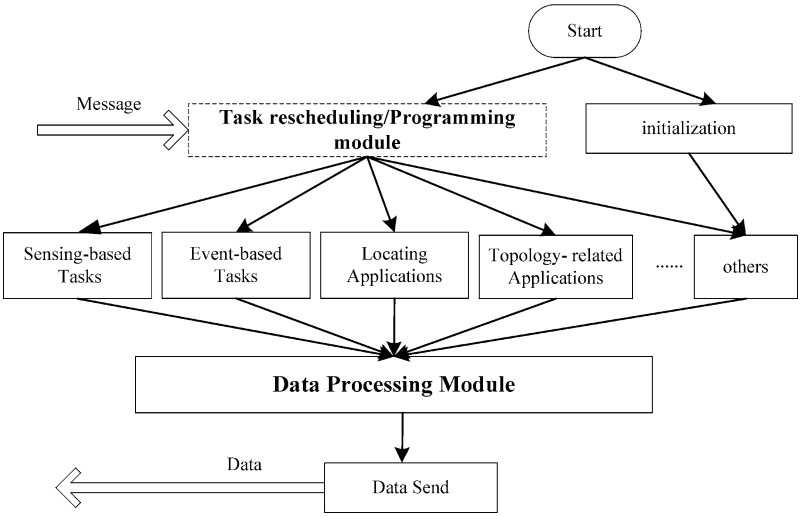

Fig. 7 Workflow of task operating

As requirements of applications are various, tasks must cooperate with the components of M-MAT to accomplish the work. The diagram shown in Fig. 6 indicates the cooperation in entire system.

In task rescheduling/programming module, nodes receive the messages from gateway and users, and then decide if the messages are stored in cache according to the priority of it. High priority messages are dealt with at once and general messages are cached to wait to be handled. If the messages belong to node-level, Task Manager(TM) treats with the messages received directly, such as start the application tasks by requirement of messages. Application-level messages will be forwarded to the relevant application tasks.

By default, some basic application tasks, such as Locating Application, are started when system boots on node devices. If there is no received messages that application tasks need to be reconfigure or modified, they collect and send sensed data to gateway and users in the light of default sampling frequency. WSNs node may also work in sleep state, which there is no any application task run in node. When receiving the messages that node need to start, stop or reconfigure some tasks, application tasks may adjust themselves according to the requirement. Meanwhile, task manager can do some operation based on the relative messages. The workflow is shown in Fig. 7.

5 Conclusion

With the deployment and actual application of Internet of things, sensor network middleware has become an increasingly important factor that determines the cost and efficiency of application development. To alleviate the develop difficulties and enable a fast and flexible deployment of IoT related industrial application, we present a middleware architecture which adopt a design principle of "Ends to the middle". Our proposed architecture considers the actual requirements of sensor network applications development and clarifies relationships between different services under various scenarios. We implement the architecture in java language. The actual application of our proposed middleware platform demonstrate that the implementation is highly modular and efficient, offers good performance even under high loads in different scenarios.

Acknowledgements This work was supported in part by grants from the National Natural Science Foundation of China (Grant # 60803131) and The Science and Technology Plan of Shenyang (Grant # 1091176-1-00).

References

1. Buratti C., & Conti, A. (2009). An overview on wireless sensor networks technology and evolution. *Sensors*, *2009*(9), 6869–6896.
2. Yu, B., & Prasanna, K. V. K. (2004). Issues in designing middleware for sensor networks. *IEEE Network*, *18*(1), 15–21.
3. Salem, H., & Nader, M. (2006). Middleware for wireless sensor networks: a survey. In *Proceeding of the first international conference on communication system software and middleware (COMSWARE 2006)*, New Delhi, India, January. New York: IEEE
4. Wang, M. M., Cao, J. N., Li, J., et al. (2008). Middleware for wireless sensor networks: a survey. *Journal of Computer Science and Technology*, *23*(3), 305–326.
5. Khedo, K. K., & Subramanian, R. K. (2009). A service-oriented component-based middleware architecture for wireless sensor networks. *International Journal of Computer Science and Network Security*, *9*(3).
6. Ramakrishnan, S., & Emary, I. M. (2011). Classification brain MR images through a fuzzy multiwavelets based GMM and probabilistic neural networks. *Telecommunications Systems*, *46*(3), 245–252.
7. Hong, S. R. W. (2005). TinyDB: an acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems*, *30*(1), 122–173.
8. Zapata, M. G., Zilan, R., Ordinas, J. M. B., Bicakci, K., & Tavli, B. The future of security in wireless multimedia sensor networks A position paper. *Telecommunication Systems*, *45*(1), 77–91.

9. Ville, S. L., & Dickman, P. (2003). Garnet, middleware architecture for distributing data streams originating in wireless sensor networks. In *Proceedings of 23rd international conference on distributed computing systems workshops*, Providence, RI, 19–23 May 2003.
10. Aberer, K., Hauswirth, M., & Salehi, A. (2006). *The global sensor networks middleware for efficient and flexible deployment and interconnection of sensor networks* (Tech. rep. LSIR-REPORT-2006-006).
11. Liu, T., & Impala, M. M. (2003). A middleware system for managing autonomic, parallel sensor systems. In *Proceedings PPoPP'03*, San Diego, CA, USA, June 2003 (pp. 107–118).

Yingyou Wen Liaoning Province, China. Birthdate: March, 1974. He received computer science Ph.D. in Northeastern University in China, and did his post-Ph.D. research in Neusoft Group. And his research interests are mainly on wireless communication and sensor network.
He is an associate professor of College of Information Science and Engineering, Northeastern University, China.

Zhi Li Liaoning Province, China. Birthdate: July, 1980. He is computer science B.S., graduated from College of Information Science and Engineering, Northeastern University, China.
He is a graduate student of College of Information Science and Engineering, Northeastern University. And research interests on wireless sensor network.