Extended Architecture for Agilla Middleware

Pradeep N
M.Tech, IV Semester,
Dept. of Computer Science & Engineering,
AMC Engineering College, Bangalore

Nirmala S
Associate Professor,
Dept of Computer Science & Engineering,
AMC Engineering College, Bangalore.

***Abstract*: Wireless sensor networks have a wide range of applications which is becoming attractive to researchers and industries. Middleware is distributed software which enables multiple applications to work on a distributed network. This paper focuses on the general issues in the middleware for WSN. It also examines the various approaches of middleware design, compares and suggests different types of applications where each approach can be used. In this paper, we propose an extended version of Agilla middleware to resolve the issue of message loss in the existing Agilla middleware.. We propose a packet delivery based routing algorithm to ensure guaranteed delivery of messages.**

Keywords: WSN, Middleware, Agilla, Distributed networks.

I. INTRODUCTION

Wireless sensor network refers to a gathering of spatially disseminated self-governing sensors to monitor physical conditions such as temperature, sound, and so on. WSN supports a wide range of applications like object tracking, infrastructure monitoring, habitat monitoring, battle field monitoring, health care monitoring etc. WSNs have found many applications in different areas, including environmental surveillance, intelligent building, health monitoring, intelligent transportations, etc.

Developing applications for WSN is a tedious job as the application developers have to meet considerable number of constraints due to the rigid integration of sensor nodes to the physical world. Designing a middleware is a novel approach for addressing these constraints wherein the middleware can act as binding software between applications and operating systems (OS). Middleware is a software that acts as a bridge between an operating system or database and applications.

The inspiration driving the exploration on WSN middleware derives from the gap between the abnormal state necessities from pervasive processing applications and the varied nature of the operations in the hidden WSNs. The application requirements include high flexibility, re-usability, and reliability. The complication of the operations with a WSN is characterized by constrained resources, dynamic network topology, and low level embedded OS APIs.

WSN middleware helps the programmer develop applications in several ways. First, it gives fitting framework contemplations, so that the application programmer can concentrate on the application logic without thinking excessively about the lower level implementation details. Second, it gives code reuse administrations, for example, code redesign, and information administrations, information separating, so that the application developer can convey and execute the application without being messed with complex functions. Third, it helps the programmer in system base administration and adjustment by giving productive asset administrations, e.g., power administration. It provides framework integration and also framework security.

Despite the fact that middleware is a well established research area in circulated processing frameworks, WSN postures new challenges to middleware research. The conventional middleware procedures cannot be connected straightforwardly to WSNs. First, most distributed system middleware techniques provides transparency abstractions by concealing the context information, but WSN-based applications should usually be context-aware. Third, data aggregation in intermediate nodes of the forwarding path is desirable in a WSN, but no such kind of support is provided in traditional distributed system middleware because of the end-to-end paradigm used. Finally, WSN requires the middleware to be light weight. for usage in sensor nodes with constrained processing and vitality assets.

II. RELATED WORK

In recent years, a new wave of networks labelled Wireless Sensor Networks (WSNs) has attracted a great deal of considerations from specialists in both scholastic and mechanical groups. WSNs can be used to form the underlying sensing and network infrastructure for pervasive computing environments [Wang, M., Cao, J., Li,J., and Sajal K. Dasi. (2006)]. A WSN comprises a gruop of sensor nodes and a sink node joined through remote channels, and can be utilized to manufacture appropriated frameworks for information gathering and handling, covering the elements of on-field sign sensing and preparing, in-system information conglomeration, and self-composed remote communication. WSNs have discovered numerous applications in diverse territories, including natural observation, smart building, health monitoring, intelligent transportations, etc.

This survey paper is concerned with middleware for WSNs. Middleware refers to programming and instruments

that can help conceal the many-sided quality and heterogeneity of the fundamental equipment and system stages, facilitate the administration of framework assets, and build the consistency of applications executions. WSN middleware is a sort of middleware that provides desired services to sensing based pervasive processing applications that make utilization of a wireless sensor network and the related installed working framework or firmware of the sensor nodes.

Wireless sensor network comprise of a great deal of sensor nodes, which are minimal effort, little size and low power supply. This sort of systems has a few preferences than conventional system, for example, simple to send, wide versatility, portability, simple to use in diverse complex situations for some special purpose. For instance, [Md.Atiqur.R.(2009)] WSNs can be widely used for environment monitoring, structure health monitoring, object tracking system, industrial process control and so on. Because of the absence of structure and asset, WSNs additionally have some restriction compared to conventional system. For example, complicated structure topology when sensor nodes are added or removed; overhead communications needed between nodes and gateway; limitation of node physical resources. Middleware solutions are developed to solve those problems in application. There are lots of existing middleware solutions with different features and mechanisms.

Developing applications for WSN is a tedious job as the application developers have to meet considerable number of constraints due to the rigid integration of sensor nodes to the physical world. Designing a middleware is a novel approach for addressing these constraints where in the middleware can act as binding software between applications and operating systems (OS).

III. PROPOSED SOLUTION

Our solution for increasing the speed for accessing tuple request/reply message is based on the popularity of tuple message access request by the sink node. The message here refers to the agent migration message. The main reason for delay in accessing tuple messages is that request message travels the path to the final node.

Increase in speed of access of tuple messages is ensured by replicating messages based on the popularity of tuple messages which travels in the path of the nodes. The following process ensures the increase in speed of access of tuple messages by the sink node.

1. Tuple popularity module.
2. Decision to replicate.

Tuple popularity module replicates requested tuple message to the sensor node based on the popularity of tuple message access request by the sink node. Therefore the tuple message can be accessed faster by the sink node. To improve the speed of access of tuple message, we propose popularity based replication algorithm is used. The algorithm notices the popularity of tuple message access request and replicates the messages in the path of the destination so that it can be accessed with lesser delay by the sink node. The algorithm works as follows.

a. When any tuple request pass through the node, the count of tuple access is kept in that node.

b. When the reply pass through the node, how many number of hops the tuple is away is recorded in the intermediate nodes.

c. When the number of access goes above a threshold which is calculated based on both access count and the number of hops away, the decision is made in the node to replicate and store the tuple.

d. By this way, the speed of access of tuple after replication is reduced, since the request and reply has to travel only few hops.

e. This mechanism is energy efficient, since the number of request / replies in network is reduced.

Decision to replicate module is used to decide whether to replicate the messages at the sensor nodes in the path of the destination. The decision to replicate the tuple messages at the sensor nodes is taken based on popularity of tuple message access computed in the previous module.

To increase the speed of access of tuple messages we propose the following architecture change in Agilla middleware.

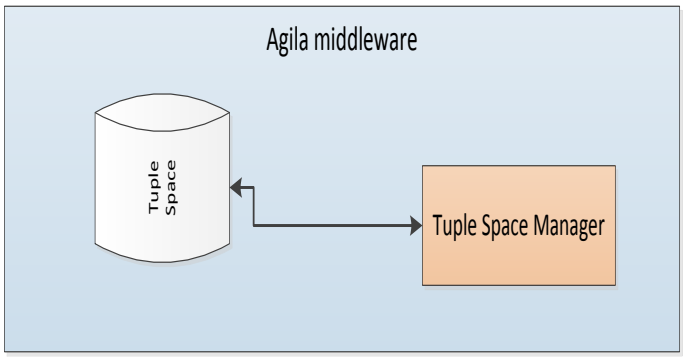

Fig 1. Agilla Middleware.

IV. REFERENCE MODEL OF WSN MIDDLEWARE

A. *Model Overview*

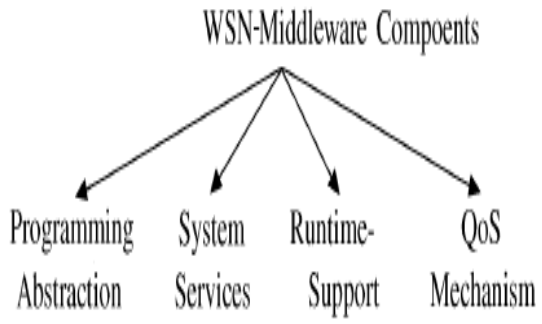

Fig.2. Major components of WSN-middleware.

As shown in Fig.2, the WSN-middleware solution should include four major components: programming abstractions, system services, runtime support, and QoS mechanisms. Programming abstractions define the interface of the middleware to the application programmer. System services provide implementations to achieve the abstractions. Runtime support serves as an augmentation of the implanted working framework to support the middleware administrations. QoS mechanisms define the QoS constrains of the system.

By analyzing the necessities of WSN-based applications and the qualities of WSNs, we propose a reference system to depict the association and connections of the above components. It should be mentioned that it is redundant for a particular WSN-middleware to incorporate all the components. Also, elements of a few parts may be consolidated together and executed as one part.

In the deployment, the elements of WSN middleware can be disseminated to the sensor nodes, the sink nodes, and abnormal state application terminals, as shown in Fig.3. The distributed middleware components situated in distinctive nodes of the system correspond with one another to accomplish some regular objectives.

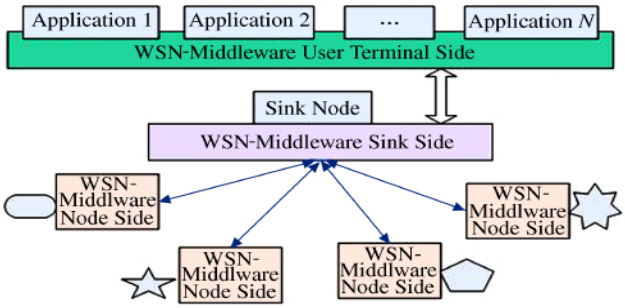

Fig.3. System architecture of WSN-middleware.

B. *Programming Abstractions*

Programming abstractions are the foundation of WSN-middleware. They provide the high-level programming interfaces to the application programmer, which separate the development of WSN-based applications from the operations in the underlying WSN infrastructures. They also provide the basis of developing the desirable middleware services. Three aspects are involved in developing the programming abstractions: abstraction level, programming paradigm, and interface type.

Abstraction Level refers to how the application programmer views the system. Node level abstraction abstracts the WSN as a distributed system consisting of a collection of sensor nodes, and provides the programmer the support for programming the individual sensor nodes for their actions and cooperation.

Programming paradigm refers to the model of programming the applications. It is often dependent on the applications. WSN applications can be grouped in two measurements: application information accumulation highlight and application element highlight. Information accumulations can be persistent, occasion driven, or question based. Application can be absolutely static and has some portability trademark, for example, versatile target or portable sink.

Interface type refers to the style of the programming interface. As a matter of fact, programming abstraction is embodied as the programming interface. Descriptive interfaces give SQL-like dialects for information question, guideline based decisive dialects for order execution, or XML-based detail records for connection design.

C. *System Services*

System services embody the functionalities and structure the center of WSN-middleware. They are presented to the application programmer through the abstraction interface, and give the backing to application deployment, execution, and also sensor and system administration. We group the framework administrations into two general classifications: common services and domain services.

Common services are the fundamental administrations imparted by all WSN applications. They help deal with the application data and the WSN foundation. The functionalities provided by the common services include:

- Code Management: responsible for code migrating and code updating in a deployed network.
- Data Management: responsible for data acquisition, data storage, data synchronization, data analysis, and data mining.
- Resource Discovery: responsible for discovering newly joined sensor nodes and detecting nodes, which are becoming inaccessible either as a result of mobility or loss of battery power.
- Resource Management: responsible for managing the node resources (e.g., energy, memory, A/D device, communication module) and network resource (e.g., topology, routing, system time).
- Integration: responsible for integrating WSN and its applications into other networks, such as the Internet and Grid, for broader use.

Domain services encourage the advancement of applications in a particular area. They can make utilization of

the regular administrations and include application-oriented functions of providing domain specific services. For example, EnviroTrack is a WSN middleware that backings ecological Target following. Impala is a middleware for the ZetbraBet project, a wildlife monitoring project. It has two layers: the upper layer contains the application particular conventions and capacities, and the lower layer contains the basic administrations, for example, code administration. WSN-SHM middleware is intended for creating structural health monitoring applications which have the prerequisites of high recurrence examining and high asset utilization.

D. Runtime Support

Runtime support gives the hidden execution environment of applications and can be seen as an augmentation of the embedded operating system which gives functions of scheduling of tasks, inter-process communication (IPC), memory control, and power control in terms of voltage scaling and component activation and inactivation. The requirement for runtime support in WSN middleware originates from the truths that the hardware and firmware of the sensor nodes may not generally give enough support to the usage of the middleware administrations.

E. QoS Mechanism

Ouality of service (QoS) mechanism is an advanced feature of WSN-middleware. Providing QoS support in WSN is still an open issue for research. QoS highlights are continually intersection layers and intersection parts, and are embodied epitomized in different functional services. For example, the information management service is required to be reliable and of high precision.

Typical parameters for expressing QoS of WSN system framework incorporate message delay, jitter, and misfortune, system data transmission, throughput, and latency. Common parameters for communicating QoS of WSN applications incorporate information precision, aggregation delay, scope, and system lifetime. Middleware acts as a broker between the applications and the system base. QoS support may decipher and control the QoS measurements between the application level and the system level. In the event that the QoS prerequisites from an application are not practical to satisfy in the system, the middleware may negoitiate another QoS guarantee with both the application and the system.

V. PERFORMANCE ANALYSIS

We implemented the proposed algorithm for increasing speed for access of tuple messages based on tuple popularity of messages accessed by the sink node. We compared the average delay of tuple messages for Agilla middleware and proposed Agilla middleware.

This performance graph shows comparison of average delay between Agilla and Improved Agilla for 100 nodes.

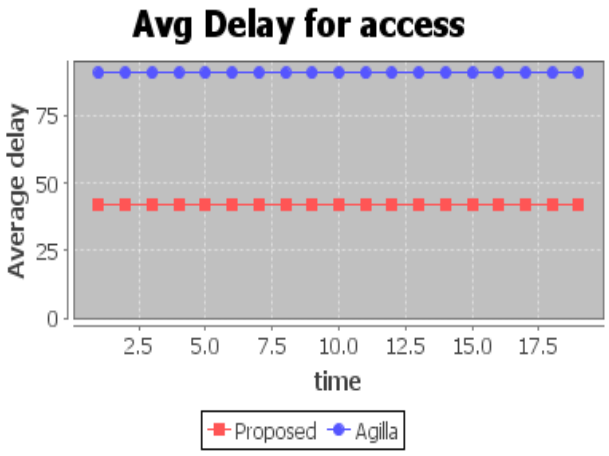

Fig. 4 Performance graph

From the above graph it is inferred that the proposed Agilla middleware has a lesser delay than existing Agilla middleware.

VI. CONCLUSION

In this paper, we have presented a review of the state of arts of middleware for wireless sensor networks. We described a reference model for WSN middleware architecture as a basis for our discussion. We discussed and evaluated the issues in the current Agilla middleware. Using the proposed Agilla middleware, we resolved the issues of existing Agilla middleware. The performance results shows that the proposed Agilla middleware has a better performance than existing Agilla middleware.

REFERENCES

[1] Wang, M., Cao, J., Li,J., and Sajal K. Dasi. (2006). " Middleware for Wireless Sensor Networks: A Survey" .Department of Computer Science, Internets and Mobile Computing Lab, Department of Computer Science and Engineering,, Supported by Hong Kong Polytechnic University under the ICRG grant NO.G-YE57, Hong Kong RGC under the Grant of A Research Center Ubiquitous Computing and the National Hi-Tech Research and Development 863 Program of China under Grant No.2006AA01Z231.

[2] Md.Atiqur.R.(2009)."Middleware for wireless sensor networks Challenges and Approaches". TKK T-110.5190 Seminar on Internetworking.

[3] Tong, S. (2009)." An Evaluation Framework for middleware approaches on Wireless Sensor Networks " . Seminar on Internetworking. Helsinki University of Technology, TKK T-110.5190.

[4] Molla M.M., Ahamed S.I. 2006. A survey of middleware for sensor network and challenges. In Proceedings of IEEE International Conference on Parallel Processing Workshops (Milwaukee, Wisconsin, August 14 – 15, 2006)

[5] Yao, Y. and Gehrke, J. 2002. The cougar approach to innetwork query processing in sensor networks. SIGMOD Rec. 31, 3 (Sep. 2002), 918.

[6] Yu, X., Niyogi K., Mehrotra, S. and Venkatasubramanian N. 2003. Adaptive Middleware for Distributed Sensor Environments. IEEE DS Online 4, 5 (May 2003).

[7] Madden, S. R., Franklin, M. J., Hellerstein, J. M., and Hong, W. 2005. TinyDB: an acquisitional query processing system for sensor networks. ACM Trans. Database Syst. 30, 1 (Mar. 2005), 122173.

[8] MC Lin, YC Chen, SL Tsao. (2012). "Design and implementation of a [8]home and building gateway with integration of nonintrusive load monitoring meters " Industrial Technology (ICIT), 2012 IEEE International Conference. 148-153.

[9] Prasannasrinivasa.S and Suman.N. (2012). "A Study of Middleware for Wireless Sensor Networks". International Journal of Research and Reviews in Ad Hoc Networks (IJRRAN) Vol. 2, No. 2, June 2012, ISSN: 2046-5106.
[10] Kumar.A, Xie.B. (2012)." Handbook of Mobile Systems Applications and Services. "University of Louisville, Louisville, KY, USA, University of Cincinnati, Cincinnati, OH, USA.
[11] Md.Atiqur.R. (2009). "Middleware for wireless sensor networks Challenges and Approaches". Helsinki University of Technology, Seminar on Internetworking. TKK T-110.5190.
[12] Fok, C.-L. , Roman, G.-C. , and Lu, C. (2009). “Agilla: A mobile agent middleware for self-adaptive wireless sensor networks" .ACM Trans. Autonom. Adapt. Syst. 4, 3, Article 16 (July 2009).