Evaluation of Middleware Architectures and Desirable Characteristics Design for Adaptive Middleware In RFID Application

Supervisor : Dr. Siti Zaiton Mohd Hashim

Mardiyono
Faculty of Computer Science & Information Systems
81310 UTM Skudai
Johor Darul Takzim
Universiti Teknologi Malaysia
Email: mardiyono@polines.ac.id
Phone : 0167415211

ABSTRACT

Middleware is a layer of the software that connects and manages application components running on distributed hosts. Several types of middleware are reflective middleware, event-based middleware, object-oriented middleware, and message-oriented middleware. Adaptive middleware is part of reflective middleware, that allows the system's behavior to be altered at run time to better match the system's current operating environment (e.g. whether the system is being debugged or operating in a real-time environment or in a secure environment). Many vendors develop the middleware as distributed software. Sun, IBM, CORBA, Microsoft and Sybase provide its own architecture in many purposes such as in communication, manufactory, medical, and financial application. IBM offered the RFID infrastructure which had adaptive characteristics. It supported multiple RFID reader devices and minimized network overload. In Radio Frequency Identification (RFID) application in particular, middleware is implemented for increasing the system performance in network communication, coordination, reliability, scalability, and heterogeneity. RFID application is used to process data sent to and received from radio frequency identification (RFID) devices such as RFID readers, writers, and printers. Non adaptive system can not modify its behavior in response to change in new situation and disturbances, so by design of adaptive middleware in RFID application, can increase the system's performance; more flexible, robust and can adapt to disturbances. This paper presents the detail study of existing middleware architectures and concludes with essential desirable design characteristics for adaptive middleware in RFID application.

Keywords: middleware, architecture, adaptive, RFID application

1. Introduction

Middleware is a layer of the software that connects and manages application components running on distributed hosts. In distributed application, middleware has many purposes to hide and abstracts many of complex details of distributed programming [1].

Middleware provides a robust and flexible abstraction for connecting applications together. As connectivity is the major aspect addressed in middleware, historically middleware is implemented as monolithic software targeted to specific environments and application domains. As a result, non adaptive middleware implementations can only perform satisfactorily in their target environments and often can not be used in other types of applications.

This inflexibility can usually be attributed to middleware implementations' inability to adapt the behaviours or structure of the platforms to the requirements of changing environments and application domains [2].

Recently the convergence of lower cost and increased capabilities of radio frequency identification (RFID) made businesses and researches take a hard look at what RFID can benefit [3]. RFID is the general term for technology that identifies an object using radio frequency transmission [4]. In Radio Frequency Identification (RFID)

application in particular, middle ware is used for increasing the system performance in network communication, coordination, reliability, scalability, and heterogeneity.

To adapt in many dynamic situation and disturbances, the middleware should has adaptive characteristic. It gives the ability to adapt or conform to new circumstances such as hardware change, failure, and increase in new applications.

In this paper, the extensive evaluation of existing middleware is presented and it concludes with essential desirable design characteristics for adaptive middleware in RFID application. It is organized as follows: section 2 discusses the related research. Section 3 presents the middleware architecture. Section 4 describes RFID middleware. Section 5 and 6 discuss the adaptive middleware and desirable characteristic. At the end, it concludes with summary and future work in section 7.

2. Related Research

The middleware architectures for RFID application have been proposed by many researches. RFID middleware architecture for distributed in large-scale system has proposed by Bo at. al in [8]. They designed the middleware architecture base on three layers (distributed resource management layer, data management layer and information representation layer). To process the information from the tag, they presented the middleware framework which comprises two engines named as network engine and data management engine. In network engine there is a component that is called adapter which the function is to adapt the system with many readers from different manufactures.

The second research explored about adaptive cleaning for RFID data streams [7]. The adaptive algorithm cleaned the useless RFID data streams in per tag and multi tag. It distinguished between periods of dropped readings and periods where the tag has actually left the reader's detection filed. The system is called SMURF (Statistical Smoothing for Unreliable RFID data).

The existing adaptive middleware above had adaptive characteristics in compatibility with different RFID reader and cleaned the useless RFID data stream. Other disturbance situations were not yet discussed. So we will extensive the characteristics to adapt with adding the new applications, failure situation such as hardware or software error, and configuration change. The adaptive middleware will be designed in tree functions such as detection (from error and new application), adaptation (adaptive algorithm), and rule and knowledge base.

3. Middleware Architecture

In another definition, Middleware is system software that resides between the application and the underlying operating system, network protocol stacks and hardware. It bridges the gap between application programs and the lower level hardware and software infrastructure in order to coordinate how parts of applications are connected and how they interoperate and enable, simplify the integration of components developed by multiple technology suppliers [9]. Middleware exists between network operating system and application component[1]. It is shown in fig 1.

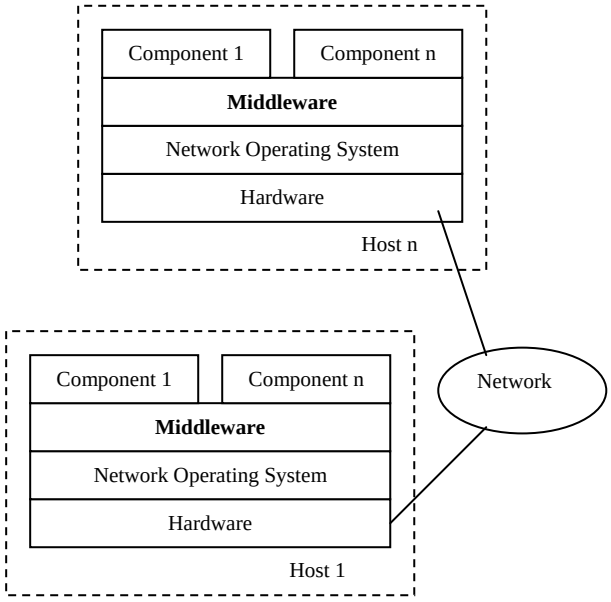

Fig 1 Middleware in distributed system

3.1 Existing Middleware

Middleware in distributed object computing composed multiple layers, such as host infrastructure middleware, distribution middleware, common middleware service, and domain specific middleware service. Distribution middleware enables client to program distributed applications much like stand alone applications [9].

3.1.1 Middleware Technologies

Three technologies will be discussed to develop RFID middleware, first CORBA,

second Java RMI and Microsoft DCOM or COM+.

3.1.1.1 Common Object Request Broker Architecture (CORBA)

CORBA is an emerging open distributed object computing infrastructure being standardized by the Object Management Group (OMG). CORBA automates many common network programming tasks such as object registration, location, and activation, request demultiplexing, framing and error-handling; parameter marshalling and demarshalling, and operation dispatching [11]. The architecture of CORBA is shown in figure 2.

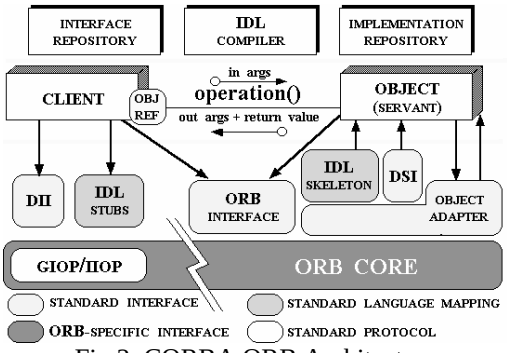

Fig 2. CORBA ORB Architecture

3.1.1.2 Sun's Java Remote Method Invocation (RMI)

The design goal for the RMI architecture was to create a Java distributed object model that integrates naturally into the Java programming language and the local object model. The RMI implementation is essentially built from three abstraction layers. The first is the Stub and Skeleton layer, which lies just beneath the view of the developer. This layer intercepts method calls made by the client to the interface reference variable and redirects these calls to a remote RMI service. The next layer is the Remote Reference Layer. This layer understands how to interpret and manage references made from clients to the remote service objects. The transport layer is based on TCP/IP connections between machines in a network. It provides basic connectivity, as well as some firewall penetration strategies [13]. This architecture is shown in fig 3:

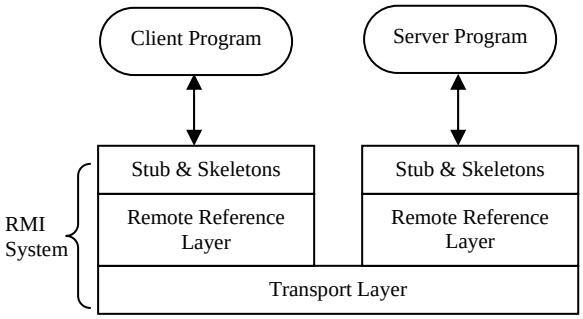

Fig 3 RMI Architecture Layer

3.1.1.3 Microsoft Distributed Component Object Model (DCOM)

The Distributed Component Object Model (DCOM) is a protocol that enables software components to communicate directly over a network in a reliable, secure, and efficient manner. DCOM is based on the Open Software Foundation's DCE-RPC spec and will work with both Java applets and ActiveX components through its use of the Component Object Model (COM). The Microsoft Distributed Component Object Model (DCOM) extends the Component Object Model (COM) to support communication among objects on different computers-on a Local Area Network (LAN), a wide area network (WAN), or even the Internet [14]. The Microsoft DCOM architecture is described in fig 4.

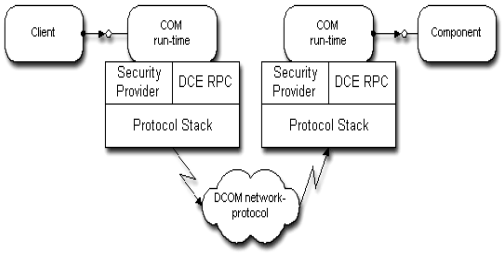

Fig 4. Microsoft DCOM Architecture

3.1.2 Middleware Comparison

According to Zarras(2004) , many parameters involved in comparison between middleware infrastructure of CORBA with J2EE and COM+ among the parameters measured are : openness, scalability, performance, access transparency, location transparency, concurrency transparency, failure transparency, migration transparency, persistence transparency, and transaction transparency. The summary of comparison of CORBA, J2EE, and COM+ are described that CORBA is good in scalability, predictability, location, failure, migration transparency.

JAVA is good in concurrency transparency, and Microsoft DCOM/COM+ is good in transaction persistence same as CORBA.

It can be concluded that CORBA, in general, provides more facilities for satisfying typical requirements imposed by distributed applications [15]. But Microsoft DCOM is more reliable in handling service interruption and able to recover in catastrophic failover test [12], meanwhile Java technology good in multiplatform that can be implemented in many operating systems.

Middleware can be used in many systems or applications in distributed computing particularly for RFID applications [5][6].

4. RFID Middleware

Before we discus RFID middleware, it's better to know the RFID technology. RFID (Radio Frequency Identification) is used to process data sent to and received from radio frequency identification (RFID) devices such as RFID readers, writers, and printers. RFID is the next generation wireless communication technology applicable to various fields such as distribution, circulation, transportation, etc. RFID is a non-contact technology that identifies objects attached with tags. Tags consist of a microchip and antenna. RFID readers obtain the information of objects and surroundings through communication with tag antennas [10]. RFID system is made up of three components: tags, readers and the host computer system.

4.1 Tags

An RFID tag is a tiny radio device that is also referred to as a transponder, smart tag, smart label or radio barcode. The tag comprises of a simple silicon microchip (typically less than half a millimeter in size) attached to a small flat aerial and mounted on a substrate. The whole device can be encapsulated in different materials (such as plastic) dependent upon its intended usage. The finished tag can be attached to an object, typically an item, box or pallet and read remotely to ascertain its identity, position or state.

Fig 5 Variety of RFID Tag Formats

4.2 Readers

The reader, sometimes called an interrogator or scanner, sends and receives RF data to and from the tag via antennas. A reader may have multiple antennas that are responsible for sending and receiving radio waves.

4.3 Host Computer

The data acquired by the readers is then passed to a host computer, which may run specialist RFID software or middleware to filter the data and route it to the correct application, to be processed into useful information.

Three primary frequency bands are being used for RFID:

a. Low Frequency (125/134KHz) – Most commonly used for access control, animal tracking and asset tracking.
b. High -Frequency (13.56 MHz) – Used where medium data rate and read ranges up to about 1.5 meters are acceptable. This frequency also has the advantage of not being susceptible to interference from the presence of water or metals.
c. Ultra High-Frequency (850 MHz to 950 MHz) – offer the longest read ranges of up to approximately 3 meters and high reading speeds.

Applying middleware architecture in RFID has been proposed by Floerkemeier at .al (2005) and Hoag at. al (2006). Floerkemeier at .al proposed middleware architecture which composed into many function such as aggregation, filtering, buffering, virtual tag memory service and messaging [5]. The architecture is shown in fig 6.

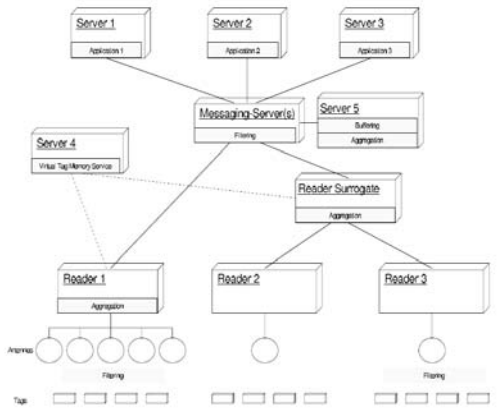

Fig 6 Middleware architecture for RFID

The diagram shows how an event-based messaging system decouples applications and readers. It also features the virtual tag memory system and the surrogate concept that address the limitation of tag memory and the heterogeneous reader landscape respectively [5].

Hoag at. al used a mixture of agent and object technology in their middleware architecture. They developed two software layers: a general-purpose core architecture and RFID-related application architecture that specializes the generic agent architecture [16]. Communication between reader agent, data base management system (DBMS) and some user interface agents in this architecture is show in fig 7.

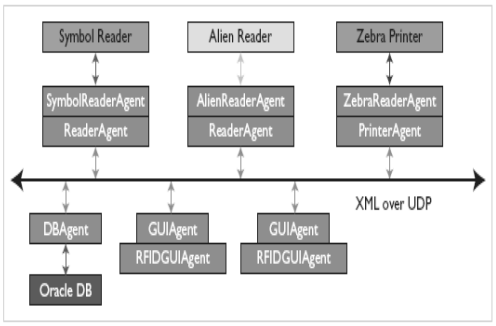

Fig 7 Reader agent-DBMS Communication

5. Adaptive Middleware

This chapter discusses about adaptive and adaptive middleware. Adaptive is from the word "to adapt" means to change a behavior to conform to new circumstances. Adaptive has been defined on the following three different levels which also imply an increasingly challenging application: [17]

a. Adaptation to a changing environment. In this case the system must adapt itself to a drifting (over time, space, etc.) environment, applying its intelligence to recognize the changes and react accordingly. This is probably the easiest concept of adaptation for which examples abound: customer's preferences in electronic commerce system, control of non stationary systems (e.g. drifting temperature), telecommunication systems with varying channel or user characteristics.

b. Adaptation to a similar setting without explicitly being ported to it. Here the accent is more on the change of the environment itself than on a drift of some features of the environment. Examples are systems that must be ported from one plant to another without a need to explicitly perform a porting of its main parameters, or a financial application that must be ported from a specific market to a similar one (e.g. a different geographical location).

c. Adaptation to new/unknown application. This level is the most futuristic one, but its open problems have been addressed already by a number of researchers, especially in the Machine Learning and Hybrid Intelligent Systems fields where, starting from very limited information about the problem it is possible to build a system through incremental learning.

Adaptive middleware architecture can control application-aware adaptation behavior and to optimize the *adaptation strategy* towards application-specific performance criteria [18]. That's why an adaptive middleware more adaptable in difference (adding new application, reader change from different manufacture) or disturbance (hardware error, disconnected) situations than traditional middleware.

6. Desirable Characteristic

The situations or disturbances which are indicated to disturb the RFID middleware: reader error, disconnected, reader change with different manufacture, and adding new application. If one of situations in above occurs, the RFID middleware may not work properly.

Base on previous research in adaptive middleware for RFID, we will add some features in adaptive characteristic:

a. Adaptable with new application.
b. Adaptable with hardware failure
c. Adaptable with new configuration

These characteristics can be provided by adaptive middleware architecture; and it will be the focus of this research and it will be

validated with several case studies application such as asset management system.

7. Conclusion and Future Work

In this paper, the middleware architectures, RFID application and desirable characteristic for adaptive RFID middleware has been studied. Adaptive characteristic can be provided by adaptive middleware to overcome/adapt the disturbance or new situation.

Further work will be carried out on proposing and developing the adaptive middleware architecture. The proposed middleware will then validated to prove the effectiveness of this desirable characteristic.

References:

[1] Jiyong Park, Saehwa Kim, Wooseok Yoo, Seongso Hong, *Designing Real T ime and Fault-Tolerant Middleware for Automotive Software*, Proceeding of SICE-ICASE International Joint Conference, 2006

[2] M. Renato, C. Renato and K. Fabio, *A middleware for Experimentation on Dynamis Adaptation*, Proceedings of the 3rd workshop on Adaptive and reflective middleware, RM'05 Nov 28- Dec 2 2005, France

[3] W. Ron, *RFID: A Technical Overview and Its Application to the Enterprise*, IT Pro, IEEE Computer Society, May/June 2005

[4] *A Basic Introduction to RFID technology and its use in the supply chain*, http//www.printronix.com/uploadedfiles/Laran_WhitePaper_RFID.pdf, January 2004

[5] Christian Floerkemeier and mathias Lampe, *RFID middleware design- addressing application requirements and RFID constraints*, Joint sOc-EUSAI conference, Grenoble, October 2005

[6] Taesu Cheong; Youngil Kim, Yongjoon Lee, *REMS and RBPTS: ALE-compliant RFID Middleware Software Platform*, Proceeding of . The 8th International Conference Advanced Communication Technology (ICACT), 2006, Volume 1, 20-22 Feb. 2006 Page(s):699 – 704

[7] Shawn R. Jeffery, Minos Garofalakis, Michael J. Franklin, *Adaptive Cleaning for RFID Data Streams*, VLDB Endowment ACM, September 2006

[8] Feng Bo, Li Jin-Tao, Zhang Ping, Guo Jun-Bo, Ding Zhen-Hua, *Study of RFID Middleware for Distributed Large-scale Systems*, Proceeding of Information and Communication Technologies (ICTTA) 2006 volume 2 page(s):2754-2759, 24-28 April 2006

[9] Richard E.Scantz, Douglas C.Schmidt, *Middleware for Distributed System Evolving the Common Structure for Network-centric Application*, http://www.cs.wustl.edu/~schmidt/PDF/middleware-chapter.pdf

[10] F. Zhou, C. Chen, D. Jin, C. Huang, and H. Min, *Evaluating and Optimizing Power Consumption of Anti-Collision Protocols for Applications in RFID Systems*, in Proc. of ACM International Symposium on Low Power Electronics and Design (ISLPED), 2004.

[11] CORBAServices Specification. OMG Document, http://www.omg.org/, accessed date:03/04/2007

[12] Middleware Company Study: .NET vs. IBM WebSphere, http://msdn2.microsoft.com/en-us/vstudio/aa700839.aspx, accessed date :March, 14 2007

[13] Remote *Method Invocation*, http://java.sun.com, accessed date:03/04/2007

[14] *The Distributed Component Object Model*, http://www.dalmatian.com/com_dcom.htm accessed date: 03-04-2007

[15] Apostolos Zarras, *A Comparison Framework for Middleware Infrastructures*, Journal of Object Technology, Vol. 3, No. 5, May-June 2004

[16] Joseph E.Hoag and Craig W. Thompson, *Architecting RFID Middleware*, Published by the IEEE Computer Society SEPTEMBER-OCTOBER 2006

[17] Bogdan Gabrys, Jens Strackeljan, Kauko Leiviska, *Do Smart Adaptive System Exist?*, Springer Berlin, 2005

[18] Baochun Li at .al , *Adaptive Middleware Architecture for a Distributed Omni-Directional Visual Tracking System* , http://www.eecg.toronto.edu/~bli/paper/mmcn00.pdf