Adaptive Middleware For The IMA

Libin Cao *
School of Reliability and Systems Engineering
Bei Hang University
BeiJing,China
290551289@qq.com
* Corresponding Author

Xing Sun
School of Reliability and Systems Engineering
Bei Hang University
BeiJing,China
foolstudent@live.cn

Abstract—With the three generations of development, avionics systems has tended to miniaturization, integration, and intelligence development. IMA is integrated and distributed that use electronic systems for commercial aviation COTS software (DRE). IMA is integrated and complex logic structure that requires the use of middleware technology to shield security and reliability issues arising from cross-platform. By studying adaptive resource management proposed for integrated avionics system we features adaptive cross-platform middleware strategy.

Keywords-IMA; Middleware; Resource management; QoS.

I. INTRODUCTION

Avionics system after three generations of development, which has tended to miniaturization, integration, intelligence development. The main features of the structure is the use of COTS components and distributed solutions and the entire avionics system integrated into a distributed system (DRE)[1]. From a software perspective, since the system functions gradually changed from the original hardware to software-based, hardware, which is supplemented by the calculation mode. Meanwhile, the complexity of the software has also brought such as cost, development time, security and many other issues. Therefore, in order to address the complexity of IMA, IMA need reduce development effort of software. Middleware technologies have been used to naturally IMA software designs, such as CORBA and TAO. However, with the development of avionics, traditional middleware is increasingly unable to meet the development of needs, mainly due to: (1) Uncertainty hardly support the load management. (2) real-time transplant too much difficulty (3) System function and the degree of coupling QoS too. To this end, traditional middleware needs to be improved and refined to meet the needs of a new generation of IMA, IMA change the current system software development too difficult dilemma.

The middleware architecture is general, making it possible to implement the middleware in any operating environment (e.g. with CORBA ORBs, Java RMI, etc.).A prototype of the architecture has been implemented for evaluation purposes in Java, but the interfaces of the components can be translated directly to CORBA IDL. In recent years, a variety of QoS architectures and tools have been developed [2]. Most of them are specifically for multimedia and telecommunications applications [9, 10, 14]. They typically allow the specification and control of application-specific quality parameters (e.g. rate, latency and jitter). Most architectures provide services to specify quality parameters of a single task (e.g., a transmission task) in an application and mechanisms for managing a single resource (e.g., network bandwidth). Other QoS management frameworks support the characterization of pre-defined fixed end-toend quality levels and provide services to request fixed quality levels or resource allocation from the environment [14, 15]. In some environments (e.g. CORBA),the same task can have multiple implementations specified in the task description [16]; these implementations may limit the available QoS options. Some works (e.g.,[13, 14]) focus on QoS management at the end-points of an application. In contrast, the networking community has been concerned primarily with providing qualities such as bandwidth, fairness, latency, etc. to flows on networks [2, 3, 10] but seldom considers the end-toend tradeoffs at the application level. The architecture described here is designed to facilitate the specification of quality of services of diverse application components and the specification and management of quality levels of individual components within a complex application in order to maintain the end-to-end quality. Components use results produced by each other and hence are dependent. Quality levels of dependent components are dependent. The negotiation algorithms provided in this architecture not only determine amounts of resource allocation but also the quality levels of individual components within an end-to-end application.

II. MIDDLEWARE

A. Definition of middleware

The so-called middleware, is located on a software layer between the operating system and application software, which provides services to a variety of application software, the application process can be different in the case of masked platform differences, communicate with each other through a network, usually in actual use, put together to form a group of middleware integration platform (including the development of platforms and operating platform), which must have a communication middleware complete the communication between the middleware. In this sense, the middleware must have the following characteristics[2]:

1) *Standard protocols and interfaces.*
2) *Distributed computing, to provide network, hardware, operating system transparency.*
3) *To meet the needs of a large number of applications.*
4) *Be able to run a variety of hardware and operating system platforms.*

B. *Adaptive Middleware*

Adaptive middleware, by definition refers to adaptive middleware inherent characteristics, can change with changes in the external environment and internal implementation of the software for the adaptive configuration and punch configurations, including the structure and function of the dynamic behavior adjustment to meet changing needs, so as to achieve the middleware, "the transformation strain" features. Adaptive middleware has the same essential characteristics of traditional middleware, provides a unified programming model for distributed application development, and other details of the underlying heterogeneity shield development and application platform. Except that: adaptive middleware is an open, dynamic, reconfigurable, intelligent and reliability characteristics[3].

III. ADAPTIVE MANAGEMENT STRATEGY

IMA is a distributed real-time environment, the need for many different types of resources for effective management, in order to complete the distributed real-time applications to share the different tasks under the distributed real-time environment, and in distributed real-time systems require multiple different distributed real-time applications, including hard real-time distributed applications and soft real-time distributed applications as non-real-time distributed applications. These real-time distributed application not only different, but have different structures, from periodic independent applications, multimedia streaming applications, parallel pipeline applications to the event-driven model of interactive applications. In addition, these applications are required to implement distributed QoS control distributed real-time applications and QoS guarantees[4]. Therefore, a new generation of real-time middleware goal is to be able to achieve unified management of resources with adaptive mechanism resource management model, and the ability to adapt to highly variable resource management needs and flexible real-time request processing. Adaptive resource management middleware, real-time distributed system must be able to promptly respond to real-time events, but also provides proactive adaptive behavior. ARTs-ARM adaptive mechanism based on several trigger events:

- New tasks and resource allocation request arrives.
- Distributed Application End Task.
- The current distributed application QoS request changes.
- The current distributed applications operating quality change.
- And the current external environment of distributed applications change.

Based on these types of distributed real-time applications, trigger events, in ARTs-ARM are specified in two different adaptive management strategies[5]:

- External adaptive management strategy: that adaptive resource management strategies affect the interaction between distributed real-time applications
- Internal adaptive management strategies: a distributed real-time applications refers to internal adaptive resources management strategy.

Based on these two adaptive resource management strategies for different situations, ARTs-ARM define different QoS adaptation mechanism. QoS control of distributed real-time computing tasks in real-time environment, a direct reflection of the real-time system operability. Because of a large number of real-time interactions between applications, increasing the difficulty of QoS control. ARTs-ARM system model QoS quantified control for defining a region for the application of acceptable QoS area. In ARTs-ARM achieve a multidimensional QoS representation, an application for each dimension specified acceptable range [Qmin, Qmax] used as control parameters QoS application. Thus, the scope of the multi-dimensional QoS defines a QoS area, this QoS model, strengthen the resource control, the realization of QoS quantitative management, and make resource management more flexible.

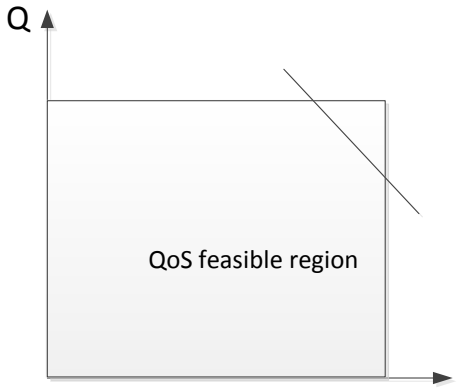

Figure 1. QoS requested region

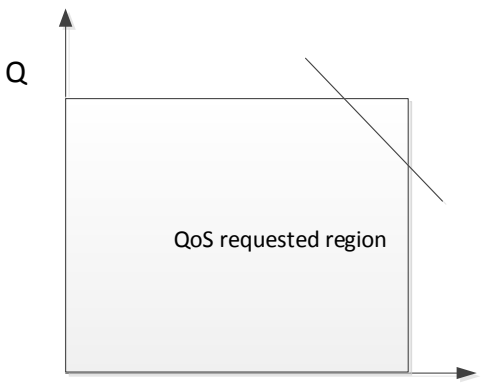

Figure 2. QoS feasible region.

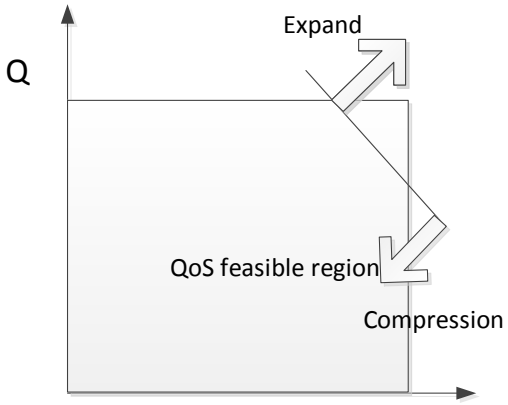

Figure 3. QoS Expand and Compression

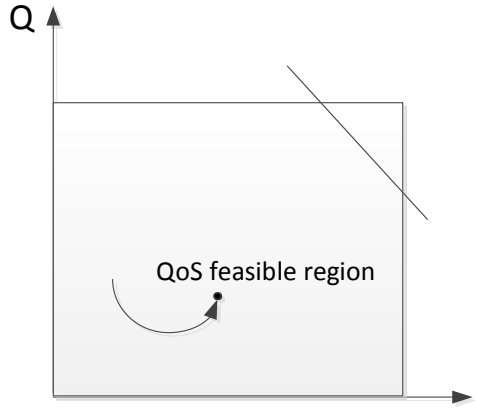

Figure 4. QoS feasible region point

In Fig. 1,2,.3 and 4,the form of two-dimensional coordinates of a general schematic representation of distributed applications and related changes in QoS area, region point represents the current execution point when QoS distributed application execution when the context changes Implementation of vertical and horizontal coordinates of the point indicates that the associated QoS value. When the distributed application execution point is called the current QoS current operating point[6].

IV. CONSTRUCTION OF IMA-ARM SYSTEM

A. *IMA-ARM design requirements*

On your CD, please indicate the format and word processor used. Please also provide your phone number, fax number and e mail address for rapid communication with the publisher. Please always send your CD along with a hard copy that must match the CD's content exactly. If you follow the foregoing, your paper will conform to the requirements of the publisher and facilitate a problem-free publication process.

In order to facilitate building a reusable and more robust application system for QoS guarantee strategy and mechanism of independent on functional component, through selection, customization and dynamic binding policy and mechanism for application system to provide performance guarantees, we will rtarmm design into a Tao middleware services based on, the rtarmm can for application of each functional component provide flexible QoS guarantee service, and independent of the realization of the functional components. Therefore, the design of RTARMM[7] considers the following factors:

- Adaptive target: that is, from which aspect to provide the system with QoS adaptive QoS to ensure that, such as: the task of real-time, network transmission delay, streaming media continuity, etc.. In this paper, the adaptive goal is to achieve the goal of real-time, through the dynamic adjustment of CPU resources to achieve.
- Adaptive strategy: for the system provides adaptive QoS ensure adopted methods and means, such as: task redistribution strategy, feedback control strategy, these strategies tend to have universal, applicable to multiple adaptive target.
- Adaptive mechanism: the basic measures for the realization of adaptive strategies, such as preemption mechanism, priority based allocation mechanism, etc.
- System status information: that is, the RTARMM in the CPU resource management system parameters, such as: CPU utilization, real-time task of the deadline miss rate, etc.
- Adaptive strategy: from the algorithm and programming to achieve a specific adaptive strategy.

B. *System structure and algorithm of QoS negotiation[8]*

The architecture of the QoS negotiation is shown in Fig. 5 The work of the QoS monitor is to collect hardware information and convert them into the standard amount of path and application, the corresponding application load and resource usage information.

The function of the analyzer is to monitor the occurrence of QoS conflicts. The development trend of QoS, load and resource utilization is calculated.[9] The cause of QoS conflict is determined by analyzing the scene. Negotiations to complete the two functions: first, it can choose the methods to solve the conflict of QoS, then put forward resource to the resource manager's request;secondly, when the Explorer is unable to perform the resource allocation scheme, negotiations can get the possibility of the highest QoS scheme. Resource allocation scheme with the highest QoS level can be determined by negotiation in the case of resource availability constraints.

Resource manager obtains the current network and computing resources from the host monitor. Then map different resource requirements to the available resources. Explorer to find the resources to meet the needs. If the resource is available, the resource manager predicts the future of QoS. Finally, the resource manager performs this heavy assignment (by program control and start and stop based on the predicted QoS values. This new selection technique can ensure that the system obtains a higher QoS level. QoS manager and explorer, respectively, to maintain QoS and management resources.[10] Whenever there is a conflict between the acquisition of sufficient resources and the guarantee of high QoS, the QoS

manager and explorer will negotiate a solution. To accomplish this work, both algorithms and communication protocols are required. Flow diagram of Fig. 6. describes the details of the algorithm and protocol.

If the resource manager does not execute the program, the QoS manager sends a negotiation message. QoS manager sends a request to the resource manager, or the next stage of the negotiation. The response of the resource manager to the QoS manager is a process of reducing the expected value of QoS. The QoS manager will evaluate whether the QoS's improvement value exceeds the limit value. If the QoS value of the improved value is achieved, the QoS manager sends a confirmation message to the resource manager.

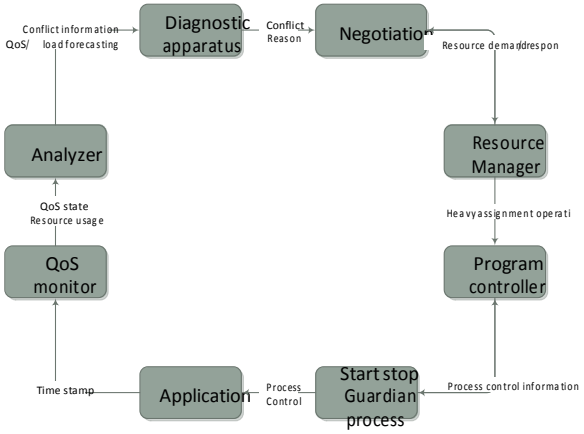

Figure 5. QoS negotiation system structure

V. CONCLUSIONS

Papers resources the IMA system dynamic service characteristics that are attributed to an adaptive distributed systems. IMA is real-time adaptive resource management mechanisms through resource management strategies ARTs-ARM introduced its location in the system, architecture, and focuses on the resources of its unified management strategy which is based on the distributed application environment, dynamic self-adaptive management mechanisms and associated resource allocation strategies. In addition, adaptive middleware software provides feedback to control task redistribution and support two adaptive strategies for the realization of uncertain load management, more over complete the mission complex aviation environment. By achieving real-time portable and flexibility to change QoS, we ensure the mechanism that provides a theoretical and technical support.

REFERENCES

[1] Zhang Wei. “common open software architecture on distributed avionics system” [D] Chengdu: University of Electronic Science and Technology, 2013.

[2] Juan Lopez Rubio."Service Oriented Architecture for Embedded(Avionics)"[D].UPC,2011.

[3] Carlos O'Ryan and Douglas C.Schmidt."The Design and Performance of a Real-time CORBA ORB Endsystem"[J].University of California, Irvine Irvine, 2000.

[4] Manuel-Díaz,Daniel Garrido ,Luis Llopis, JoséM.Troya,"Designing distributed software with RT-CORBA and SDL"[C]Computer Standards&Interfaces, 2009.

[5] Barbara Han, "Real-time distributed systems Dynamic Scheduling Service - Design and implementation of CORBA environment" [D] Chengdu: University of Electronic Science and Technology, 2004.

[6] Gao Hui-Sheng, Han Yong, He Yu-Jun, "Real-time CORBA in information collection system application" [J] Computer Applications and Software Volume 31, No. 6, June 2014.

[7] Yeh, Y. C., “Triple-Triple Redundant 777 Primary Flight Computer,” in Proc. 1996 IEEE Aerospace Applications Conference, v. 1, New York, N.Y, February 1996.

[8] Sha, L., “Using Simplicity to Control Complexity,” IEEE Software 18(4) , July/August 2001.

[9] Brower, R. W., “Lockheed F-22 Raptor,” in The Avionics Handbook, C. Spitzer, ed., Boca Raton, FL, CRC Press, 2001

[10] Moore, J., “Advanced Distributed Architectures,” in The Avionics Handbook, C. Spitzer, ed.,Boca Raton, FL, CRC Press, 2001.

[11] Klara Nahrstedt and Ralf Steinmetz. Resource Management in Networked Multimedia Systems. Computer, 28(5):52–63, May 1995.

[12] Object Management Group (OMG). Object Services: Common Object Services Specification. Technical report, OMG, 1997.

[13] D.Schmidt, A. Gokhale, T. Harrison, and G. Parulkar. A High Performance End System Architecture for Real-Time CORBA. IEEE Communications Magazine, 35(2):72–78, 1997.

[14] W. Feng and Jane Liu. Algorithms for Scheduling Real-Time Tasks with Input Error and End-to-End Deadlines. IEEE Transactions on Software Enineering, 23(2):93–106, February 1997.

[15] Lixia Zhang, Stephen Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. RSVP: A New Resource ReSerVation Protocol. IEEE Network,7(5):8–18, September 1993.